



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

**DEPLOYING CROWD-SOURCED FORMAL
VERIFICATION SYSTEMS IN A DOD NETWORK**

by

Mahmut Firuz Dumlupinar

September 2013

Thesis Advisor:
Second Reader:

Geoffrey G. Xie
Thomas Housel

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 2013	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE DEPLOYING CROWD-SOURCED FORMAL VERIFICATION SYSTEMS IN A DOD NETWORK			5. FUNDING NUMBERS	
6. AUTHOR(S) Mahmut Firuz Dumlupinar				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) Manual formal software verification is an expensive and time-consuming process. Military software is currently verified manually by highly skilled analysts. To reduce the high costs of the formal verification, DARPA started a Crowd-Sourced Formal Verification (CSFV) program that aims to include as many people as possible to participate in this verification process by embedding some of the verification logics into computer games. In this study we built a network prototype for hosting a CSFV server on a DoD network. The CSFV network prototype is designed according to the common security practices, necessary security measures against possible attacks, and the Security Technical Implementation Guides (STIGs) published by DISA to provide confidentiality, integrity and availability. Important details are presented about server operating system selections, proper usage of necessary network services, and firewall and IDS rules for efficient network security. Results from common network penetration test tools confirm that our prototype meets the necessary security requirements and can be trusted on a DoD network.				
14. SUBJECT TERMS crowd sourced formal verification, network security, cyber attacks, crowdsourcing, virtualization, cloud computing, firewalls, intrusion detection systems, network penetration test.			15. NUMBER OF PAGES 83	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

**DEPLOYING CROWD-SOURCED FORMAL VERIFICATION SYSTEMS IN A
DOD NETWORK**

Mahmut Firuz Dumlupinar
Captain, Turkish Army
B.S., Turkish Military Academy, 2004

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

**NAVAL POSTGRADUATE SCHOOL
September 2013**

Author: Mahmut Firuz Dumlupinar

Approved by: Geoffrey G. Xie
Thesis Advisor

Thomas Housel
Second Reader

Dan C. Boger
Chair, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Manual formal software verification is an expensive and time-consuming process. Military software is currently verified manually by highly skilled analysts. To reduce the high costs of the formal verification, DARPA started a Crowd-Sourced Formal Verification (CSFV) program that aims to include as many people as possible to participate in this verification process by embedding some of the verification logics into computer games. In this study we built a network prototype for hosting a CSFV server on a DoD network.

The CSFV network prototype is designed according to the common security practices, necessary security measures against possible attacks, and the Security Technical Implementation Guides (STIGs) published by DISA to provide confidentiality, integrity and availability. Important details are presented about server operating system selections, proper usage of necessary network services, and firewall and IDS rules for efficient network security. Results from common network penetration test tools confirm that our prototype meets the necessary security requirements and can be trusted on a DoD network.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	INTRODUCTION.....	1
B.	THE RESEARCH PROBLEM.....	2
1.	Problem Statement.....	2
2.	Purpose Statement	2
C.	RESEARCH QUESTIONS AND HYPOTHESIS	2
D.	THESIS ORGANIZATION.....	3
II.	BACKGROUND	5
A.	INTRODUCTION.....	5
B.	CROWDSOURCING	5
1.	A Brief History of Crowdsourcing	5
2.	Current Crowdsourcing Activities	6
3.	Linux Kernel.....	6
4.	reCAPTCHA	7
5.	Games with a Purpose (GWAP) Project.....	7
6.	Verification Games: Making Verification Fun	8
a.	Amazon Mechanical Turk	8
b.	SETI@home	9
C.	VIRTUALIZATION.....	9
1.	Virtual Machine Monitors (VMM)	10
2.	VMM Types.....	10
a.	Type I VMM.....	10
b.	Type II VMM.....	11
3.	Security Considerations in Virtualization	11
a.	Hypervisor Security.....	12
D.	CLOUD COMPUTING.....	13
1.	Characteristics of Cloud Computing	14
2.	Cloud Computing Deployment Models.....	16
E.	CLOUD COMPUTING SERVICE MODELS.....	18
F.	ENTERPRISE NETWORK SECURITY	19
1.	Network Security Concepts.....	19
2.	Security Vulnerabilities, Threats and Countermeasures	20
a.	Security Vulnerabilities.....	21
b.	Security Threats	21
c.	Security Countermeasures.....	24
III.	NETWORK DESIGN.....	27
A.	INTRODUCTION.....	27
B.	NETWORK DESIGN CONSIDERATIONS AND GUIDELINES.....	27
C.	NETWORK TOPOLOGY	28
D.	SEGMENTED DESIGN.....	30
1.	Border Router and Firewall.....	30

2.	IDS.....	31
3.	Game Servers and Database Server.....	32
4.	Reverse Proxy Server	33
E.	ADDITIONAL SECURITY MEASURES.....	34
1.	Encryption	34
2.	Authentication	34
IV.	IMPLEMENTATION	35
A.	INTRODUCTION.....	35
B.	GENERAL NETWORK INFORMATION.....	36
1.	Network Topology Implementation	36
2.	Implementation of Servers	37
3.	Running Firewall Rules.....	41
4.	Reverse Proxy Deployment	44
5.	IDS Deployment	45
6.	Data Encryption	46
7.	System Validation via Penetration Testing Tools	47
V.	CONCLUSION	51
A.	SUMMARY AND CONCLUSION	51
B.	FUTURE WORK AND CONSIDERATIONS	52
	APPENDIX.....	53
	LIST OF REFERENCES	61
	INITIAL DISTRIBUTION LIST	65

LIST OF FIGURES

Figure 1.	Type I and type II VMM (From Thomas, 2013)	11
Figure 2.	NIST visual model of cloud computing (From Damiani, 2011).....	13
Figure 3.	Hybrid cloud (From Shilovitsky, 2013).....	18
Figure 4.	Comparison of service models (From Lau, 2011)	19
Figure 5.	The security triad (From Chou, 2012)	20
Figure 6.	Distributed Denial of Service attack (From Masikos et.al., 2004)	22
Figure 7.	CSFV Network topology.	29
Figure 8.	Firewall design.....	31
Figure 9.	IDS deployment.	32
Figure 10.	Database server.	33
Figure 11.	Reverse proxy server.....	33
Figure 12.	Subnets in the CSFV Network.....	36
Figure 13.	CSFV network implementation.	37
Figure 14.	VMware ESXi vSphere Client.....	38
Figure 15.	VMware ESXi command shell.	39
Figure 16.	The SSL module for CentOS	47
Figure 17.	Attacking on the game server via LOIC.	48

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1.	Cloud benefits: efficiency and innovation (From Takai, 2012).....	15
Table 2.	Virtualized systems.	39
Table 3.	Non-virtualized Systems.	40
Table 4.	Necessary packages from CentOS repository	41
Table 5.	SNORT alerts.....	49

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

CA	Certification Authority
CSFV	Crowd Sourced Formal Verification
CSP	Cloud Service Provider
DDOS	Distributed Denial of Service
DMZ	Demilitarized Zone
DREN	Defense Research And Engineering Network
EC2	Amazon Elastic Compute Cloud
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
NAT	Network Address Translation
SQL	Structured Query Language
SSL	Secure Sockets Layer
STIG	Security Technical Implementation Guide
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
XSS	Cross-Site Scripting

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I want to express my sincere gratitude to my advisor, Professor Geoffrey G. Xie for his unlimited support and guidance, and to Charles Prince for his considerate and responsive working hours with me on the CSFV project. Also I would like to thank to Professor Thomas Housel for providing the best understanding of thesis research and being my second reader. I am fortunate to have John D. Fulp as my Network Security instructor and I appreciate his help he gave whenever I knocked on his door. Another person who helped me decrease the tension and stress with his humorous approach was Umit and I want to thank to him as well.

In addition I owe my deepest gratitude to my mother, Aycan, my father, Necati, and my sister, Zeynep, for supporting and encouraging me even at the hardest moments of my research and during the long hours in my basement lab. Without their endless love and support this study would not have been completed.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. INTRODUCTION

In 2014 cyber attacks on critical infrastructure are expected to increase significantly and consequently cause security expenditures to reach a peak of \$86 billion (Rivera, 2013). Even after all the attention devoted to this threat, cyber-related problems do not seem to be resolved. According to the task force report of the Defense Science Board, the advanced cyber threat the Department of Defense (DoD) faces is too prevalent, and critical IT structures may stop functioning in case of a sophisticated and well-resourced attack vector (Kaminski, 2012). This threat becomes even more disturbing when we realize that a malicious user can cripple the Unified Threat Management System (10 million lines of code) with a malware (125 lines of code) (Dean, 2013).

Being under the cyber threat of its opponents, the DoD has taken the necessary precautions in different areas. One of those areas is the formal verification of military software, which is used to process classified information. One to five bugs are found in a thousand lines of code of military software, and most of them are related to security vulnerabilities.

Currently formal verification is performed manually by very specialized engineers and this is a very expensive process (Dean, 2013). The Defense Advanced Research Projects Agency (DARPA) initiated a project to lower the costs of the verification process and increase the efficiency levels by harnessing the power of a crowd. Crowd Sourced Formal Verification (CSFV) aims to perform the necessary software verification by creating computer games, which are fun to play. The goal is to make the crowd play the games and have the military software verified in the background. Even though it sounds innovative to use the crowd to solve a problem, it is not a new concept. Crowdsourcing was used by DARPA before to design a next-generation combat vehicle and to reconstruct shredded documents, such as those found after military engagements (Montalbano, 2011).

DARPA plans to run its CSFV systems on the Internet—possibly using cloud infrastructure (Dean, 2013). By using Amazon Compute Cloud (EC2) systems, DARPA will use ordinary people and make them play the games, which will enable the software verification with the help of complex algorithms.

This thesis aims to design and prototype a secure network for CSFV systems, and this network will be run on a limited access intranet such as the Defense Research and Engineering Network (DREN), whereas DARPA’s initial gaming environment will exist on the Internet, open to everyone.

B. THE RESEARCH PROBLEM

1. Problem Statement

DARPA’s CSFV systems are designed to reside on the Internet, and they welcome anyone to log on and play the games. However, there is currently no network architecture to run the games on classified intranets such as DREN, the Non-classified Internet Protocol Router Network (NIPRNET) or the Secret Internet Protocol Router Network (SIPRNET).

2. Purpose Statement

The purpose of this thesis will be to design and prototype a secure network architecture for implementing Crowd Sourced Formal Verification methods on classified/unclassified networks.

C. RESEARCH QUESTIONS AND HYPOTHESIS

In this thesis these research questions will be answered and explained:

1. What are the common security threats and solutions for securing enterprise network resources?
2. What are the key benefits of virtualization, and how can virtualization be implemented on DREN while meeting security requirements?
3. What are the possible security concerns of cloud computing?
4. What are the unique security challenges for deploying CSFV on DREN?

D. THESIS ORGANIZATION

Chapter I: “Introduction” (Introductory information about formal verification issues and the need for efficient software verification)

Chapter II: “Background” (Review of definitions and details of concepts related to CSFV such as crowdsourcing, cloud computing, virtualization and enterprise network security)

Chapter III: “Network Design” (Design of optimally most secure network in regard to the potential security vulnerabilities/threats specifically for CSFV networks)

Chapter IV: “Implementation” (Creation of the CSFV network prototype)

Chapter V: “Conclusion” (Summary, future work and recommendations)

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

A. INTRODUCTION

This chapter will summarize the core concepts that will be reviewed in this thesis. In the following section basic information about crowdsourcing will be presented, including its history, current use, and vast benefits. Sections C and D will explain two necessary platforms for CSFV systems, which are virtualization and cloud-computing. Finally this chapter concludes with information about the current cyber security threats pertaining to CSFV networks.

B. CROWDSOURCING

Crowdsourcing provides a workspace to be used for a large scale of activities. These activities vary from journalism to image indexing and from language translation to entertainment (Howe, 2009). Crowdsourcing is the outcome of two words: “crowd” and “outsourcing” and is meant to accomplish something with the help of untrained, ordinary people, rather than professionals and experienced employees. The crowd may be drawn from either a large or small population. The activity, which is done by the crowd may help for projects such as designing a task, developing a technology, solving an algorithm or classifying, collecting or analyzing large amounts of data (Bell, 2010). Nowadays typical examples of crowdsourcing are created online, but the first examples of crowdsourcing were quite different. Organizations have leveraged crowdsourcing solutions throughout history.

1. A Brief History of Crowdsourcing

The history of crowdsourcing dates back to 1714 when the Longitude Contest was organized by the English government. The purpose of this contest was to enable the government to find a prototype of a navigation device, which might even be developed by a common citizen, to help sailors navigate easily (Lynch, 2010). Another early example is the Toyota Logo Contest in 1936. Twenty-seven thousand people attended this event, and the best design created by someone in this crowd was chosen to be the

Toyota Logo. In 1955 the Sydney Opera House was also designed and built as a result of a public contest, which encouraged ordinary people from 32 countries to help this design project (Lynch, 2010).

2. Current Crowdsourcing Activities

Crowdsourcing provides opportunities to solve the chronic problems worldwide that have been waiting to be solved for years. The diverse base of knowledge and abilities is unlimited in a crowd. The technique of crowdsourcing would match this diverse base with the needs of people using this technique (Howe, 2009). Currently crowdsourcing activities around the world are increasing rapidly. Most IT companies are handing their technical support elements to forums in which users share knowledge. In journalism, sector leaders like Reuters and the BBC are increasingly choosing public resources to crowdsource important work, such as investigating government wrongdoings or reports about local events. In March 2009 public opinion was largely gathered and analyzed through the White House website, and those results greatly affected political decisions (Howe, 2009).

Another area of crowdsourcing is the freelance sector. Although it is a local Chinese firm, the freelancing website Zhubajie.com has more than seven million subscribers, and it is still growing (Lynch, 2012). Such growth is hardly surprising. It is a fact that a large population helps a lot in crowdsourcing. In the following subsections some globally known crowdsourcing examples will be presented along with some local examples.

3. Linux Kernel

Open source software development projects inherently allow users to see, change or add code freely to the already developed software. In the 1990s this open source development environment enabled the creation of an important product, Linux. Its creator, Linus Torvalds, declared that the operating system that he had developed was open to any critics, and he was looking for others who could help improve it (Howe, 2009). By getting the crowd's help, today Linux is in use everywhere—from supercomputers to hand-held devices.

4. reCAPTCHA

Originated at Carnegie Mellon University by Professor Luis Von Ahn, reCAPTCHA is designed to digitize the text in the books. It has been used to digitize *The New York Times* archives. Another use of reCAPTCHA is to protect websites from bots that are designed to infiltrate the restricted areas in the network (Bell, 2010).

reCAPTCHA provides websites with the images of words that bot software is unable to read. The registered websites help verify these images and present them as CAPTCHA words, and the words are sent to digitization projects. This service is calculated to provide 12,000 man-hours per day of free labor (Bell, 2010). Popular websites such as Facebook, Twitter and Ticketmaster are effectively using this service in the customer validation process (Bell, 2010).

5. Games with a Purpose (GWAP) Project

Another application in the vast area of crowdsourcing is using simple games to achieve a purpose that is much more valuable than just relaxing. Because billions of people including children are interested in games, and maybe millions of them are spending several hours daily playing games, scientists thought that these valuable hours could be used beneficially in parallel with the fun part. Carnegie Mellon University Professor Luis Von Ahn created the GWAP project to use game playing hours for a scientifically valuable purpose, such as Internet image indexing, monitoring security cameras and performing language translation (Ahn, 2006). Other possible application areas could be IT security, Internet accessibility, adult content filtering, and web search (Ahn, 2006). The main idea behind this project is to have people play games and use the results of the games for another purpose. One of the most popular of these games, the ESP game designed by Ahn, is designed for two players. These players must agree on the best word that represents the image presented to them. They do this without knowing that they are seeing the same image (Bell, 2010).

The concept of playing games is applied to other areas as well. A game such as “Foldit,” which was designed by researchers from the University of Washington, allows users to play with protein-like structures by folding and unfolding them. When the

protein structure is modified games scores are automatically recorded with regard to the success level of how accurate the folding is (Bell, 2010).

6. Verification Games: Making Verification Fun

This project is directly related to the CSFV concept, and it aims to make people play games as a part of the CSFV project. Ernst (2012) investigated ways to lower the costs of software verification by developing game-playing-based verification systems. In his latest work he introduces and explains the technique he used in the game “Pipe Jam,” which is used to map a software and to correct potential problems in that software.

His game is comprised of boards, levels and worlds in which the gamer plays with pipes that are linked to each other. The gamer tries to pass a ball into different sizes of pipes. The width of a pipe represents the type of variable that the pipe represents. A wide pipe stands for a variable that is permitted to contain a “null” value whereas a narrow pipe represents a variable that is guaranteed to be non-null.

As with the general concept of CSFV, a gamer does not have to know anything about the software he helps to verify. Gamers do not have to have confidentiality privileges, and they can be anyone from public.

The Pipe Jam game is analogous to the dataflow network for a program. It maps the source code’s type flow properties into a network of pipes. Essentially, this system converts the software into a game that can be played by ordinary people. After the player finishes a portion of the game the final board configurations can be translated into a proof of correctness for the original program (Ernst, 2012).

a. Amazon Mechanical Turk

Amazon Mechanical Turk is a platform where organizations find employees for their projects. Amazon Web Services provide this platform to match the job requesters with the potential workers who can be anywhere around the world. Job requesters post the jobs to be done on the Amazon Mechanical Turk web page and let the workers choose one or more to complete for a monetary payment (Bell, 2010).

b. SETI@home

Another example of Internet supported crowdsourcing, SETI@home was founded in 1999 and aims to use the computing power of the computers in a crowd to find any proof of extraterrestrial life. Scientists managing SETI@home regard the project supported by the crowd more important than their supercomputers because of the computing power within the crowd (Howe, 2009).

C. VIRTUALIZATION

Obviously crowdsourcing and virtualization are different academic disciplines, and it can be a little puzzling for the reader to jump directly from crowdsourcing to virtualization. However because of this thesis' interdisciplinary nature these two areas of study need to be explicitly stated here.

Virtualization is the abstraction of computing resources such as processing power, storage and network bandwidth. It helps simulate software or hardware and creates a simulated environment, which is known as virtual machine (Scarfone, Souppaya and Hoffman, 2011). Virtualized environments can have one or more of these goals:

- To allow any device connected to a network to access any network-enabled application, even if the device and the application were not designed to work compatibly.
- To isolate two or more applications to provide security or to maintain network resources.
- To isolate an application from the operating system to work with any other version of the operating system.
- To increase the number of users an application can be used for support by running the application on different instances of the operating system.
- To optimize the usage of a system by decreasing the time system resources remain idle.
- To increase system availability via redundancy by guiding the user to a running system if the previous system fails (Kusnetzky, 2011).

After a brief introduction to virtualization and its use for data centers, an overview of the types of virtualization to be used throughout the CSFV project will be provided in this section.

1. Virtual Machine Monitors (VMM)

VMM is a piece of software that gives the abstraction between machine hardware and the virtual machine. VMM records every activity happening within the limits of the virtual machine. It provides resources when necessary, and it can also forbid the usage of resources.

2. VMM Types

To virtualize an operating system, all instructions should be executed by hardware, software or a combination of both. These combinations have formed different VMM models. The models presented in this thesis will be Type I and Type II VMM models.

a. Type I VMM

This type runs directly on the machine hardware; that is why this type of VMM is called “bare metal.” A Type I VMM would most likely be an operating system or kernel that can support virtual machines (Figure 1). It would perform scheduling and resource allocation for each virtual machine on the system. Processors should be in compliance with every virtualization requirement that is needed by Type I. This compliance should provide the necessary protection for the real system from virtual machine borne intrusions (DoD ESX Server Guide, 2008).

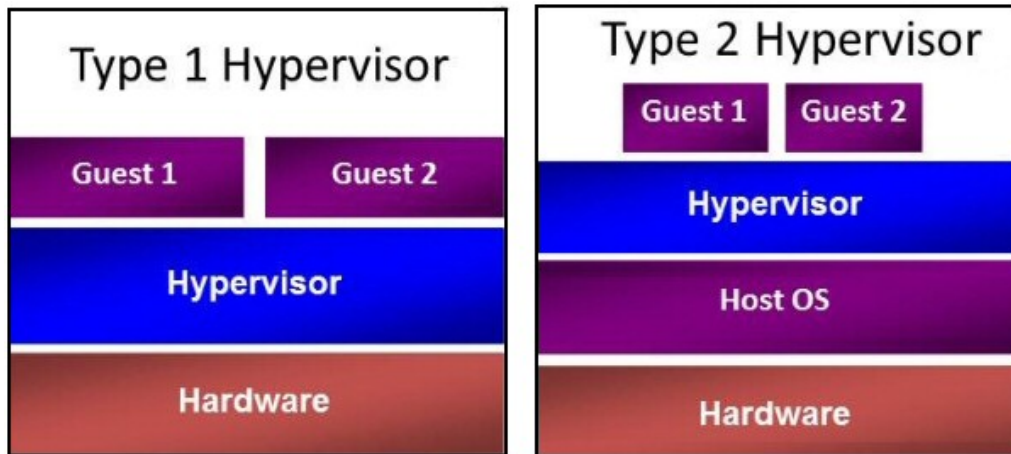


Figure 1. Type I and type II VMM (From Thomas, 2013)

b. Type II VMM

This type of VMM runs on a host operating system and is limited to the operating system resources, such as memory management, processor scheduling, resource allocation and hardware drivers (Figure 1). Because of these dependencies any operating system-related security issue might affect the stability of the Type II VMM (DoD ESX Server Guide, 2008).

3. Security Considerations in Virtualization

The overall security of a virtualization solution is dependent on the security level of each of the components. These components could be the hypervisor, host computer, host operating system (OS), guest OSs, applications on the system and storage. Virtualization users should be sure about the security levels of these elements by taking appropriate measures such as controlling access to administrator privileges, having up-to-date software, performing monitoring and analysis of logs, using anti-malware software, using host-based firewalls and any other mechanism to prevent possible attacks.

Having these security measures for virtualization may not suffice to secure a system, however, because the virtualization needs of organizations change in every situation, and each situation requires different security approaches. Here, we will dig deeper into these security approaches to clarify this concept and explain more about hypervisor security.

a. Hypervisor Security

The hypervisor known as Virtual Machine Monitor needs to be secured using methods similar to those used to protect other software. The overall security of the virtualized system is directly linked to the virtualized management system. Such systems should be under absolute control of the administrators. For example when the administrator needs to connect to the virtualized system, remote access to administration interfaces should be restricted by a firewall. If the communication is carried by an untrusted network, data should be under encryption using Federal Information Processing Standard (FIPS) approved methods (Scarfone, Souppaya and Hoffman, 2011).

Access should be limited to the hypervisor, especially if it is a bare-metal type hypervisor. Although most of the bare-metal hypervisor access methods are based on user name and password, some of them still offer additional methods that grant access to the hypervisor management interface.

Unlike bare-metal hypervisors, hosted virtualization environments do not generally have access controls. Because of this lack of security measures, anyone who can install an application on the OS can manipulate the hypervisor. This vulnerability necessitates the policies for organizations specifying the privilege level of accessing the hypervisor. The following guiding rules can help implement the right policy for hypervisors.

- Install all updates for the hypervisor as soon as they are released.
- Protect network communications via authentication and encryption using FIPS 140-2 cryptographic modules.
- Synchronize the virtualized environment to a trusted time server.
- Remove all unused hardware connected to a host system.
- Do not use hypervisor file-sharing systems if they are not needed. The file-sharing systems are considered possible attack vectors (Scarfone, Souppaya and Hoffman, 2011).

Additionally, hosted virtualization means that more threats will be around the system because the hosting OS will possibly have some vulnerabilities in addition to hypervisor security concerns. Unnecessary applications on the host OS should be

removed because the security of each guest OS will be affected by the host OS security (Scarfone, Souppaya and Hoffman, 2011).

So far the biggest concern of security personnel is hiding hypervisors from the eyes of attackers. However, hypervisors have certain characteristics that make attackers aware of their existence. The hypervisor's interactions with file systems, registry and related virtual drives all give some information to potential attackers. To prevent such attacks against a hypervisor using virtualized systems, organizations should take into consideration the risks and vulnerabilities (Scarfone, Souppaya and Hoffman, 2011).

D. CLOUD COMPUTING

According to the National Institute of Standards and Technology (NIST) *Definition of Cloud Computing* (Mell and Grance, 2009), cloud computing is a robust and dependable pool of network resources, which includes computing, storage, applications and databases. One of the most important characteristics of this pool is its rapidly releasable and on-demand nature. This system would require the least management effort and less service provider interaction (Mell and Grace, 2009). A graphic representation of the NIST cloud computing model is shown in Figure 2.

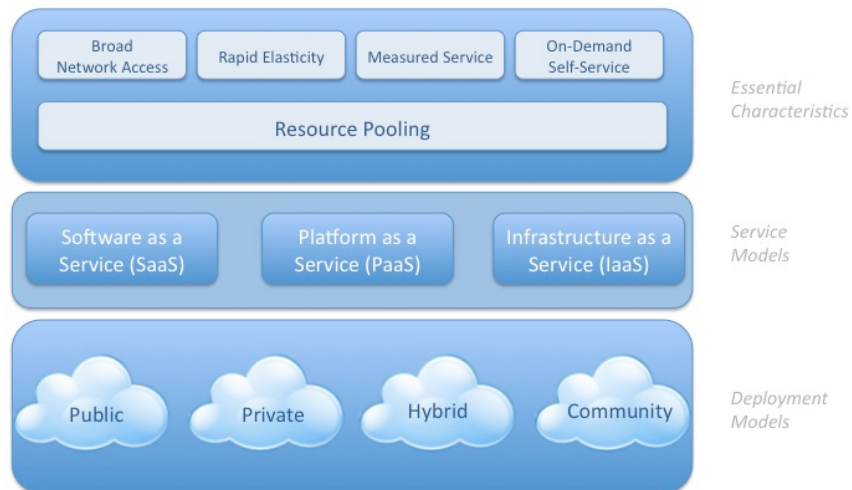


Figure 2. NIST visual model of cloud computing (From Damiani, 2011)

1. Characteristics of Cloud Computing

According to the *NIST Definition of Cloud Computing*, characteristics of this computing include on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service. These characteristics make cloud computing more beneficial in terms of efficiency and innovation in comparison to the current computing environment as shown in Table 1.

Cloud Benefits	Current Environment
<ul style="list-style-type: none"> • Improved asset utilization (server utilization > 60- 70%) • Aggregated demand and accelerated system consolidation (e.g., Federal Data Center Consolidation initiative) • Improved productivity in application development, application management, network, and end-user devices 	<ul style="list-style-type: none"> • Low asset utilization (server utilization < 30% typical) • Fragmented demand and duplicative systems • Difficult to manage systems
Cloud Benefits	Current Environment
<ul style="list-style-type: none"> • Shift focus from asset ownership to service management • Tap into private sector innovation • Encourage entrepreneurial culture • Improve link to emerging technologies (e.g., devices) 	<ul style="list-style-type: none"> • Burdened by asset management • De-coupled from private sector innovation engines • Risk-averse culture

Table 1. Cloud benefits: efficiency and innovation (From Takai, 2012)

Several cloud computing benefits are detailed in the following paragraphs.

On-demand self-service: The user can alter unilaterally the system capabilities such as computing, server and storage settings, if needed, and there is no need for interaction with a Cloud Service Provider (CSP).

Broad Network Access: Cloud capabilities are available for a wide variety of thick and thin clients through standard access methods. Those clients could be mobile phones, laptops, netbooks, tablet computers or personal digital assistants (PDAs) (Smoot and Tan, 2012).

Resource pooling: End-user needs are the main factor that dynamically assigns and reassigns the computing sources of the CSP. Those resources could include storage, processing power, memory, network bandwidth and virtual machines. The CSP has

relative freedom in notifying the end-user of the actual physical locations of provided resources. End-users are thought to be able to access these resources through an intranet if they are internal users and through the Internet if they are outsiders (Smoot and Tan, 2012).

Rapid Elasticity: The CSP can quickly change the system capabilities to both scale in and out according to the user needs. End-users mostly think that system capabilities to provision are unlimited and usable (Mell and Grance, 2009).

Measured Service: Cloud computing capabilities are under the control and surveillance of the CSP, with the help of a measuring system that often operates on a pay-per-use basis. This system provides transparency of used service (storage, computing, bandwidth) for both the cloud provider and the end-user (Mell and Grance, 2009).

2. Cloud Computing Deployment Models

There are four deployment models of cloud computing:

Public Cloud: This type of cloud is available to almost anyone in the crowd who can access the Internet. With the development of cloud technologies, service providers operating in this area are increased. Some widely known examples include Amazon's Elastic Compute Cloud (EC2), Rackspace's Cloud Offerings, and IBM's BlueCloud (Winkler, 2011). While these providers primarily offer Infrastructure service, there are others that give Application layer service, such as Google's AppEngine and Windows' Azure Services platform.

From a security perspective public clouds can be considered both secure and insecure. They are considered secure because public clouds are mainly operated by large scale CSPs. Therefore they should have enough security measures, including access control, data ownership and encryption (Winkler, 2011). On the other hand, end-users leave their data in the hands of the provider not knowing whether it is secure or not. CSPs have no obligation to their customers regarding the location of the stored data. If data is stored offshore in another country, the data is expected to be subject to the laws of the hosting country (Winkler, 2011).

Private Clouds: Private clouds are hosted internally and basically serve only one organization. Unlike data on public clouds, data on private clouds are not mixed with external users' data. However, organizations may want to provide data isolation to satisfy the needs of the organization's own subunits (Winkler, 2011). Private clouds can be owned, maintained and operated by either the organization itself, a third party or a combination of both (Mell and Grance, 2009).

From a security perspective private clouds have more constraints than public clouds because small scale organizations may not address the computation needs as do large scale CSPs operating public clouds. It would also be incorrect to assume that private clouds are more secure than public ones (Winkler, 2011). Considering that private clouds use virtualization to save more on computing resources, private cloud providers should obtain measures, such as hypervisor and virtual machine security, to secure the virtualization environment (Winkler, 2011). The point of operating private clouds is that we can address the security issues by ourselves and are free to use any further measures that we deem appropriate. We have the chance to implement the security architecture according to organizational needs. In this sense a private DoD cloud can employ stricter security measures than a private cloud that is owned by a business corporation.

Community Clouds: The cloud infrastructure is created for the use of multiple independent organizations that have the same concerns (that is, security, mission, regulation, policy or compliance). The system may be owned and maintained by each of the organizations, a third party or a combination of them (Mell and Grance, 2011). This model presents a valuable opportunity for the organizational entities that have similar legal and compliance restrictions. Different levels of community clouds are being considered both by the governments of the United States and the European Union. Governments will benefit because inter-government business transactions are considered to be processes in a possessed environment and will cause no additional costs as does the Internet (Winkler, 2011).

Hybrid Clouds: This cloud infrastructure can be a combination of two or more different cloud types, as shown in Figure 3. Here separate and different clouds remain as a unique entity, and each of them is linked to others via a technology that enables secure

data transmission (Mell and Grance, 2011). Hybrid clouds are generally preferred by entities that operate private clouds. The main reason for this preference could be either security related or financial. An entity such as the DoD can have its confidential data in the private cloud and store unclassified data in the public cloud. Here hybrid cloud infrastructures will allow the necessary data transfer of the organization (Winkler, 2011).

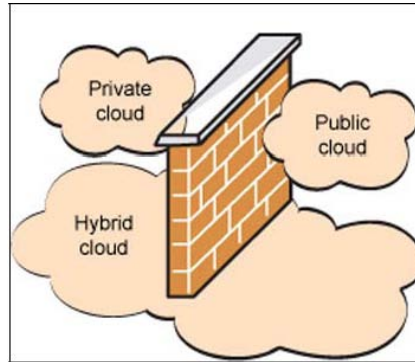


Figure 3. Hybrid cloud (From Shilovitsky, 2013)

E. CLOUD COMPUTING SERVICE MODELS

Software as a Service (SaaS): The applications running on a CSP's infrastructure are the services provided to the consumer. Those applications could either be accessed via thin clients, such as a web browser, or via the interface of software. The consumer has no privilege to change any settings of infrastructure, servers, operating systems, or storage except for some limited application configurations (Mell and Grance, 2011). Moreover, consumers may not want to change those settings. Google's GMAIL or Yahoo mail services can be considered examples of SaaS (Winkler, 2011).

Platform as a Service (PaaS): The consumer is capable of using his own application or an application provided by the CSP, as well as the libraries, services and tools (Mell and Grance, 2011). The consumer has control only over the applications he uses. He does not and cannot control the infrastructure, operating systems or servers provided by the CSP. In this sense, the PaaS is similar to the SaaS model. However, the PaaS model is different because the consumer owns the application. Google App Engine could be an example of the PaaS model (Winkler, 2011).

Infrastructure as a Service (IaaS): This service model has the most flexible components provided to the consumer. In this model consumers can change applications, storage components, operating systems, databases and even some limited networking entities such as host firewalls (Mell and Grance, 2011). Some CSPs, including Amazon, go further providing services that the consumer can access through a platform of routers, switches and data centers (Winkler, 2011). This model is compared to other cloud computing models in Figure 4.

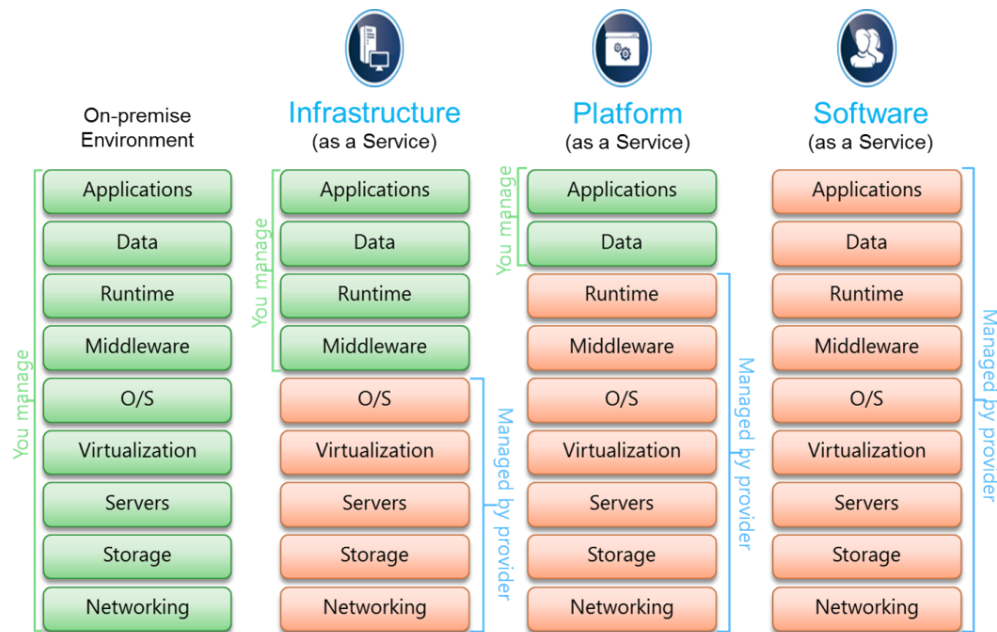


Figure 4. Comparison of service models (From Lau, 2011)

From a security perspective the IaaS model is preferable for an organization like the DoD. No matter which cloud type the DoD utilizes, the necessary solution seems to be the IaaS model.

F. ENTERPRISE NETWORK SECURITY

1. Network Security Concepts

The term “network security” is derived from “computer security.” According to NIST, computer security comprises the necessary protection mechanisms to provide

confidentiality, integrity and availability of data being processed (Stallings and Brown, 2008). These three terms are widely known as the CIA Triad, and they lay out the essential principles concerning data and information security. The CIA Triad, shown in Figure 6, refers to the confidentiality, integrity and availability of data. Confidentiality avoids the unnecessary disclosure of information to unauthorized parties. Integrity protects information so that it cannot be changed in an unauthorized manner. Availability makes sure that information is always available for authorized users and keeps the system in service (Stallings and Brown, 2008).

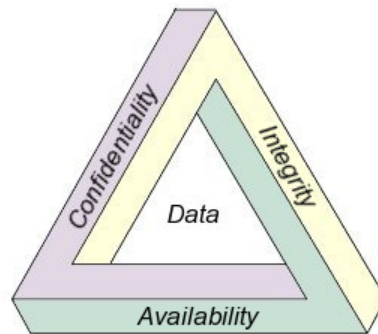


Figure 5. The security triad (From Chou, 2012)

2. Security Vulnerabilities, Threats and Countermeasures

As former Secretary of Defense Leon Panetta noted repeatedly, the next Pearl Harbor is expected to happen soon, but this time from the cyber domain (Panetta, 2012). Current Secretary of Defense Chuck Hagel also drew attention to the importance of cyber security from a global perspective (Hagel, May 2013). Indeed, providing cyber security is firstly a macro level necessity. According to the *Advanced Cyber Threat Report* cyber threats share almost the same importance level as nuclear armament issues (Defense Science Board, 2013). In this thesis study we will use an enterprise level approach and analyze vulnerabilities, possible threats and necessary cyber countermeasures to mitigate the security risks related to cloud computing.

a. Security Vulnerabilities

In the security context, when we refer to security vulnerabilities we mean the vulnerabilities of system resources such as computing power and storage. This resource could be damaged or changed in such a way that it differs from what it is supposed to be. The resource may be leaky, giving information to unauthorized parties. Also the resource could be unavailable or very slow, and it may not serve system users as expected (Stallings and Brown, 2008).

Unnecessary open ports: An open port is used for communication between computer systems on a network. A designated software works on a single port (e.g., Mail service works on port 25 on most computers). When a port is open it continuously listens for possible communications intended for it. If the service running on the port is unnecessary that port should be closed to decrease the vulnerability level.

Unpatched systems: System security vulnerabilities exist in almost all computer systems and software. They are supposed to be patched by the vendors to avoid vulnerabilities because hackers/attackers look for holes in such systems to exploit them via malicious codes (Gregory, 2010). Most software vendors react quickly to distribute patches to fix the security holes, and system users are expected to apply those patches. When necessary patches are not installed on the system, it becomes vulnerable to possible threats.

b. Security Threats

A security threat is the possibility of an adverse condition affecting computer systems. This threat could be realized from outside the enterprise or even from within an enterprise by a disgruntled employee (Gregory, 2010).

Denial of Service (DoS): The DoS prevents or limits the intended use of communication infrastructures. An attacker can scale his attack to a limited level such as the recording of security audits. Sometimes he aims to take the whole network down by disabling it or overloading the network with random messages to downgrade performance levels (Stallings and Brown, 2008).

A developed form of DoS attack is the Distributed Denial of Service (DDoS) attack, which targets network resources by overwhelming traffic. DDoS attacks could originate from thousands, or even hundreds of thousands of systems. Highly complicated DDoS attacks use a botnet (see Figure 7), which is a collection of zombie computers, controlled by botnet operators (Gregory, 2010).

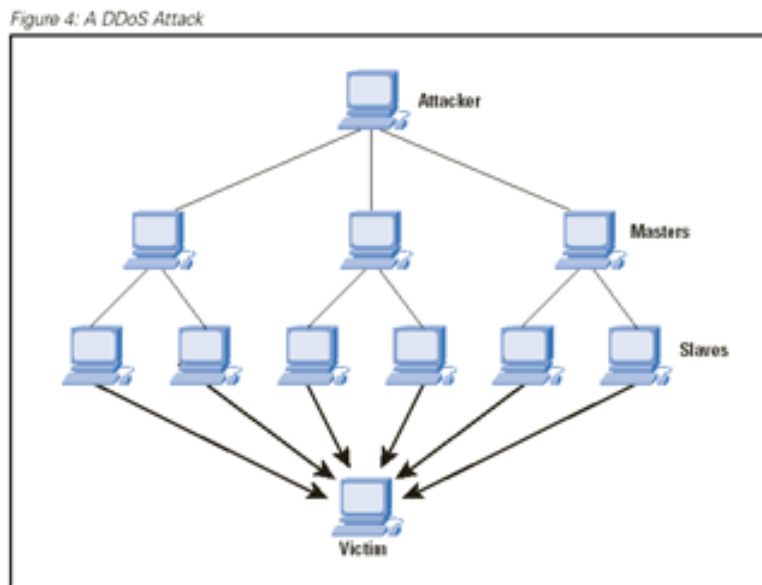


Figure 6. Distributed Denial of Service attack (From Masikos et.al., 2004)

Sequence Number: Sequence number attacks attempt to hijack or fail a TCP session between two parties by guessing the sequence number of any of the TCP packets and achieving a correct timing. The attacker then sends false packets to either one of the parties pretending to be a valid sender.

Smurf: A smurf attack includes a large number of fake Internet Control Message Protocol (ICMP) echo requests. The ICMP packets are sent to the broadcasting address of the target network, causing all the devices to respond with ICMP packets as well. The attacker changes the “from” part of the packets to the target system’s IP address. When all of the devices send replies to the echo request the target system gets overloaded (Gregory, 2010).

Spam: Spam attacks comprise a high volume of emails which mostly have commercial origins. Spam on the Internet is estimated to account for 90% of all email communication. Spam aims to degrade the performance of network devices by evading frequently used spam filters (Gregory, 2010).

Phishing: Phishing is a type of spam, and it is performed by sending mails while masquerading as official parties such as banks, hospitals or government agencies. After defrauding the mail recipients, the attacker gathers some important personal information like credit card or social security numbers (Gregory, 2010).

SQL (Structured Query Language) Injection: SQL injection attacks are one of the most significant threats to websites and databases. This type of attack mainly aims to introduce some malicious input, such as SQL codes, to a website and then gather confidential or other sensitive information from the linked database. The basic cause of SQL injection attacks is insufficient input validation measures (Halfond and Orso, 2005). This type of attack succeeds when input from a website visitor is accepted as a database SQL query without being validated. The attacker then becomes able to perform his SQL query embedded in the input for the website. In this way the intruder can gather, modify and delete data in the database. Clearly it is a threat for all three domains of information security (i.e., confidentiality, integrity and availability). In our proposed CSFV project to be run on DREN, the CSFV website could be vulnerable to this kind of attack because of the nature of its web-based applications.

Cross Site Scripting (XSS): An XSS threat resembles the previous threat of SQL injection in the way that a website lacks security measures to check the input coming from users. XSS targets the website as content defacement or DDoS attacks, whereas SQL injection aims to manipulate the database behind the website (Ernst, 2009). According to past research XSS attacks moved to the top of the cyber threat assessment in documents such as “SANS Top 25 Most Dangerous Software Errors” and Open Web Application Security Project (OWASP) lists, passing the famous buffer overflow attacks. Basically XSS attacks use special characters when giving input to Hyper Text Mark-up Language (HTML) documents such as adding <script> to inputs to invoke the JavaScripts interpreter. When the browser does not perform input validation, the attacker

becomes successful and finds ways to further exploit the website such as account hijacking, cookie poisoning and even Denial of Service (Shar and Tan, 2012).

c. Security Countermeasures

Encryption composes the essential part of network security and security countermeasures. It includes two main pieces of information security: Symmetric Key Encryption and Asymmetric Key Encryption. In short, symmetric encryption means having a single key for each cryptographic algorithm, whereas asymmetric encryption uses two different keys, one of which would be known by public (Diffie and Helman, 1976). The other key is supposed to be kept secret by its owner and would be used to decrypt the messages that were previously encrypted by the other public key. Key distribution in symmetric encryption is known to be easy because there is only a single key to be controlled by two users. Although key distribution is more difficult in asymmetric ciphers, it is more secure. The common feature of both symmetric and asymmetric ciphers is that the algorithm would be known by everyone, but the key is supposed to be known just by owner as explained in Kerckhoffs's principle (Kerckhoffs, 1883).

The network security concept also includes practical applications such as IPsec technology, firewalls and authentication (Tanenbaum, 2003). Some of these applications will be used as system security measures in the experimentation sections of this thesis.

IPsec (IP Security): Being originally a communications security measure, IPsec was created to fill the security gap throughout Internet. The argument at the first point was to provide data security either end-to-end fashion or just on the network with unaware users. After long discussions among security experts a security model emerged, and it was designed to provide network level security (Tanenbaum, 2003). IPsec has two modes of operation, which are transport mode and tunnel mode. The advantage of tunnel mode is that it adds another IP layer onto the packet, making data transfer easier and concealing flow of data in a better way.

Firewalls: The need for firewalls emerged after fast improvement of networks by means of connectivity and speed. Especially when networks overcame the size of premises and started forming Wide Area Networks (WAN) with the need for better connectivity, network input-output controls became more important. Then firewalls started to be used (Stallings, 2008). Essentially a firewall is composed of two routers working as packet filters and an application gateway. The point here is to force all of the traffic—incoming and outgoing—to use the route through the firewall. This way the firewall will allow or prohibit certain IP packets depending on whether the packets are authorized or not. This allow-or-prohibit decision is made by using IP numbers and port numbers. For instance, firewalls should prohibit data traffic coming to port 23, which is a telnet port (Tanenbaum, 2003).

Intrusion Prevention Systems (IPS): An IPS is a network-based Intrusion Detection System (IDS) with an additional capability of dropping packets and blocking traffic as well as detecting malicious traffic and sounding alarms. IPSs can either be in a form of host based or network based (Stallings, 2008).

A host-based IPS can check packets depending on their signatures or by using heuristics. Signature checking is controlling the payloads coming with the packets. The drop or allow decision is made depending on the presence of malicious content. When heuristics are used, the IPS looks for anomalies and misbehaviors of packets which can be either a modification of system resources, privilege-escalation exploits, buffer-overflow exploits or access to email contact lists (Stallings, 2008).

A network-based IPS is an inline device that has the capability of inspecting Transmission Control Protocol (TCP) packets by tearing them down. It applies this inspection on every incoming data flow, and when a malicious behavior is seen, all the future data packets pertaining to that data flow are dropped. Some of the techniques used by network-based IPS systems to find malicious packets are pattern matching, stateful matching and protocol anomaly (Stallings, 2008). An example of network-based IPS is SNORT, which is an open-source, network-based, intrusion prevention system. It can employ signature-based, protocol-based and anomaly-based inspection.

THIS PAGE INTENTIONALLY LEFT BLANK

III. NETWORK DESIGN

A. INTRODUCTION

In this chapter a secure CSFV network design will be discussed, key network design questions will be asked, potential security vulnerabilities and threats specifically relevant to CSFV networks will be presented and the necessary network components for optimal system security will be explained.

Before examining the network design, it is important to understand the following facts about the surrounding environment:

- The games and the database infrastructure are not yet released at the time this chapter is being written, so the design principles in this document will refer to presentations, drafts or other kinds of documents related to CSFV studies of CSFV vendors or DARPA.
- The network infrastructure will be focused on security design, which does not mean that performance factors of the network are totally ignored; rather they will be discussed optimally with a security emphasis.
- This system will first be deployed on the NPS Intranet and then transferred to DREN. Design decisions will be made according to Defense Information Systems Agency (DISA) Security Technical Implementation Guides (STIG).
- The necessary security measures such as the location of firewalls, usage of IPS or IDS, network segmentation for better security and specific services (reverse proxy) will be discussed in this chapter. More detailed design parameters such as operating systems, firewall-IDS rules, and virtualization technologies to be used in the CSFV network will be presented in Chapter 4.

B. NETWORK DESIGN CONSIDERATIONS AND GUIDELINES

Several network design publications were used as guidelines while creating the CSFV network. The first of those publications is written by the National Institute of Standards and Technology: NIST Special Publication (SP) 800-44, *Guidelines on Securing Public Web Servers*, explains how to operate public-facing web servers for organizations such as the DoD and the private sector. It presents general recommendations about web servers, including operating system (OS) choice,

virtualization of OSs and communication security between web servers and database servers.

CSFV networks would run on DREN-owned infrastructure, so the initial design principles should be also in compliance with the publications, which set networking rules for DREN. Network Infrastructure STIG (2007) and Web Server STIG (2006) are two publications that we used to build our network. According to these documents an optimum level of network security should be provided by the following security measures:

- DoD networks should be layered according to the information security levels that those layers contain.
- The security in-depth principle should be used in locating the network components.
- DoD system administrators should install, maintain and operate IDS inside their networks with the capability of logging.
- Firewalls and web proxies should be deployed for perimeter security.
- Web servers and database servers should reside on separate devices.
- Web servers should use Secure Sockets Layer (SSL) and Transport Layer Security (TLS).

C. NETWORK TOPOLOGY

We had three main goals before starting to design this network: to create a topology that shows the physical and logical elements of the network; to secure the network components against the most common threats of the day and from ones possible in the near future; and to prevent those security measures from degrading the throughput of the network.

There were a couple of issues in mind in the design process:

- securing the network from both Internet and possible insider attacks,
- using a multi-layered security approach to secure sensitive and classified information such as the database server,
- having additional systems for logging, encryption and intrusion detection (SANS Institute, 2003).

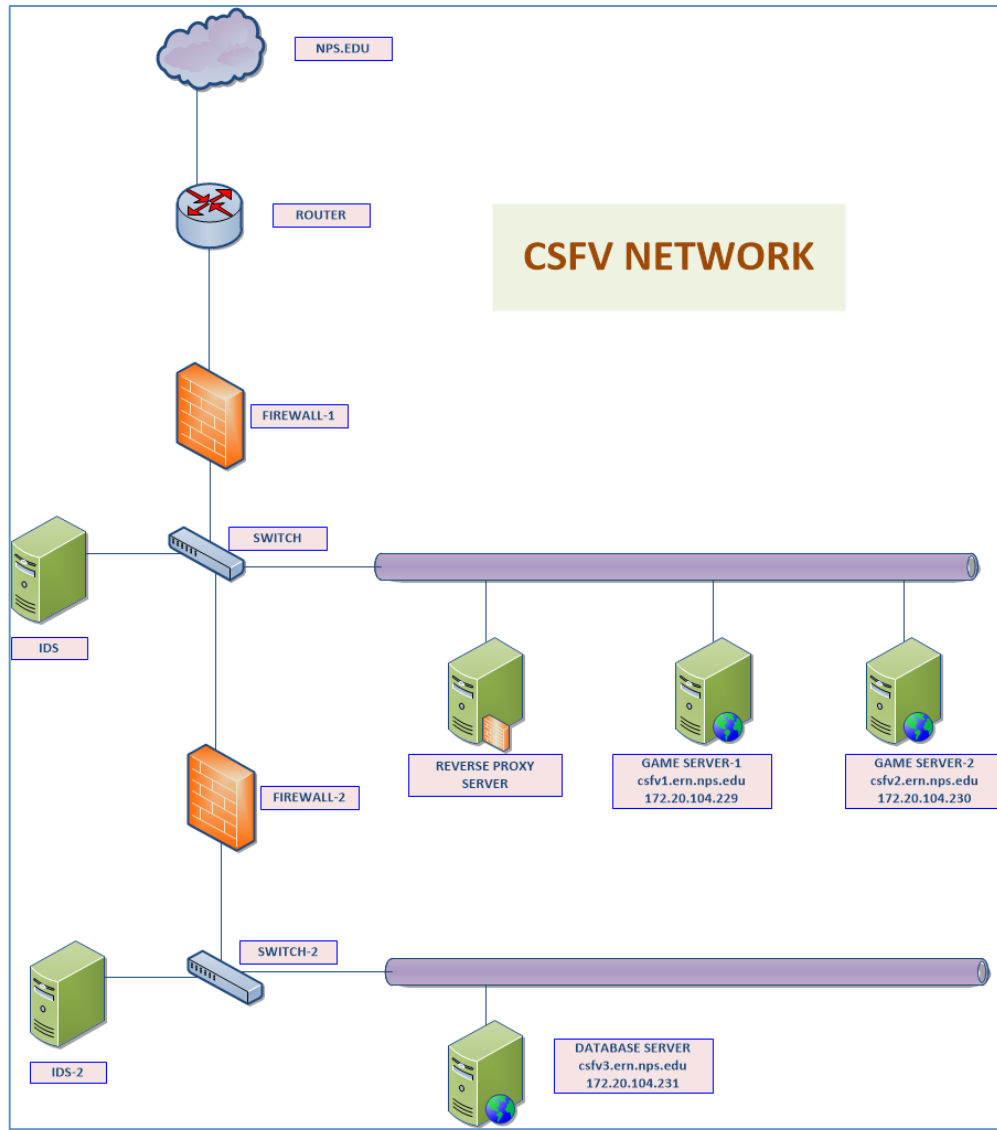


Figure 7. CSFV Network topology.

To examine the architecture in our network, we focused on the following seven questions:

- Why do we use virtualization technology for game servers and database servers?
- Why are game servers and database servers designed to reside on separate physical servers rather than being virtualized on the same machine?
- Why do we use IDS, not IPS?
- Why do we use two separate IDSs?

- What is the design principle related to using two firewalls, and what are their roles?
- What is our goal in using a reverse proxy?
- Is encryption necessary in CSFV network design, and how will it function?

D. SEGMENTED DESIGN

In this section detailed answers will be given to the preceding questions by partitioning the network (Figure 7) and explaining it part by part.

1. Border Router and Firewall

There will be a router and a firewall at the point where our network connects to the Naval Postgraduate School (NPS) website (nps.edu), as shown in Figure 7. The router and firewall can be either separate devices, or they could be designed to reside on the same physical server as a multi-functional system. The decision on this location will be given at the implementation phase.

When choosing the type of firewall, we had a couple of options. It could be a packet filter firewall, which would operate at Layer 3, a stateful inspection firewall operating at Layer 4 or a deep packet inspection firewall, which additionally inspects application level loads. A packet filter firewall would be a poor choice because this type of firewall accepts every packet without looking at the destination port. The third choice, a deep packet inspection firewall, would possibly create a choke point at the entrance of the network and decrease the throughput. A stateful inspection firewall was the optimal solution both for the IP packet inspection security and for the system performance. Choosing to use stateful inspection firewalls is also necessary in utilizing the second firewall, which comes behind the switch. The application level packet inspection job of the two firewalls would be executed by the IDSs. Again, this decision was based on system performance concerns, not to mention the high cost of deep packet inspectors or the technical difficulty in utilizing them effectively.

The first firewall will be configured to inspect packets which are coming to the reverse proxy server that will be in the demilitarized zone (DMZ), whereas the second

firewall looks into the packets coming to the database server. This second firewall only allows the packets coming from the address of game servers to database server. In case of a compromise of the reverse proxy, the attacker will not be able to access the database server because of this second firewall behind the DMZ.

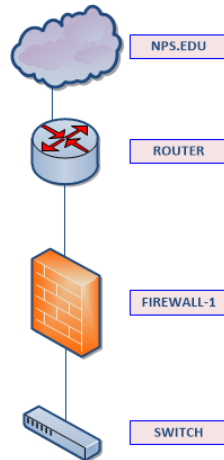


Figure 8. Firewall design.

2. IDS

To further protect the required network additional elements, in this phase we had the option of using IPS, IDS or both. When we consider that IPS would inspect a packet and sometimes necessarily drop it, we identified this possibility as a drawback for network throughput by generating false alarms (i.e., false positives). That is the reason we chose to use IDS. IDS will stay on the network and watch the ongoing traffic without interfering. It will start an alarm when malicious packets or abnormal network activity are detected. IDS will inspect the packets by comparing them to attack signatures or pre-installed IDS rules (DISA, 2007). The reason we use two separate IDS devices is to provide security in depth throughout the network. While the first IDS is inspecting the traffic in the DMZ, the second IDS will operate in a more secure part of the network and will look for anomalies or attack signatures targeting the database server.

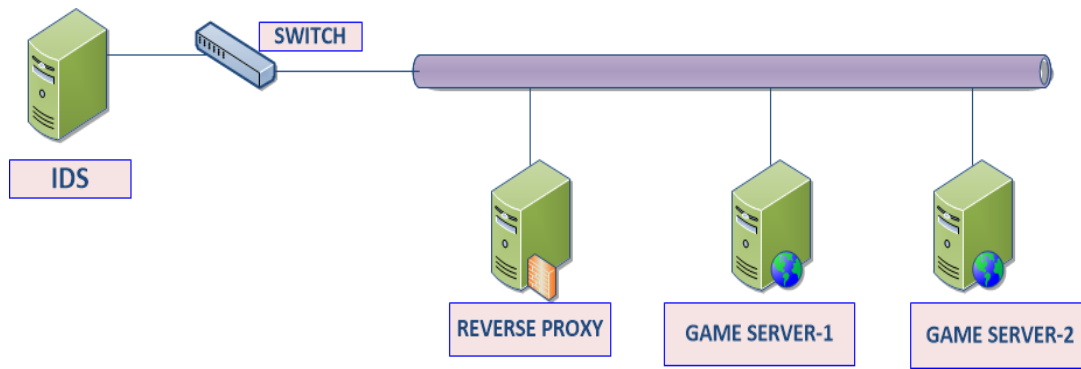


Figure 9. IDS deployment.

3. Game Servers and Database Server

Game servers in the DMZ will be virtualized on the same physical server to provide efficient resource utilization. The number of game servers will initially be two, and this initial design will test the systems in a simple infrastructure with as few negative factors as possible. Furthermore, the CSFV network would eventually be deployed on a DoD cloud (Dean, 2013), and a virtualized environment on the NPS network will help us find and solve any potential vulnerabilities related to virtualization before this final deployment.

We plan to operate the game servers in a virtualized environment; however, the database server will reside on another physical server. The reason for this design is to have multiple security layers in the network and to prevent any attacker from gaining access to our database server after compromising a game server. It is basically a two-fold security measure. Firstly, we avoid a virtualization-related “escape attack,” in which an attacker easily jumps to another virtualized system after accessing one virtualized environment. Secondly, our design makes an attack more difficult by putting the database in the trusted layer of our network.

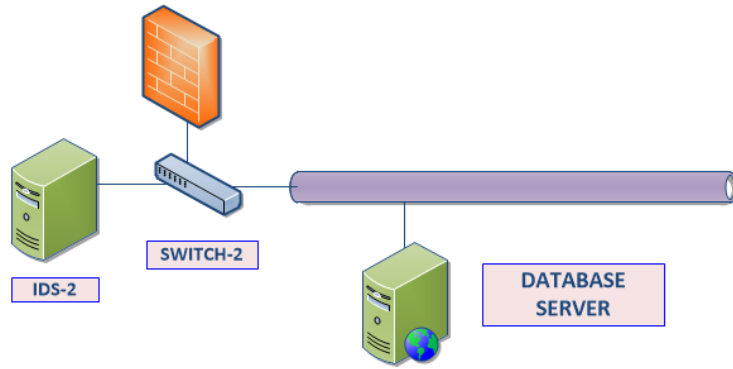


Figure 10. Database server.

4. Reverse Proxy Server

A reverse proxy will be running between the game servers and their clients. It will serve as a proxy server operating in the reverse direction. The clients of game servers will not know the IP address of the game servers; rather they will know the address of the reverse proxy server and communicate with it. Because reverse proxies are specially designed systems they are more trusted and secure than regular servers.

Additional features of reverse proxies are caching and load balancing. Caching increases the speed of the network by keeping the most frequently used records-data and retrieving them upon request. Load balancing will distribute the traffic of Hyper Text Transfer Protocol (HTTP) and Hyper Text Transfer Protocol Secure (HTTPS) over the two game servers, which will allow us to increase the number of game servers as the number of gamers expands. Moreover a load-balanced server will withstand the high volume of requests during a possible DoS attack by distributing the workload to many servers.

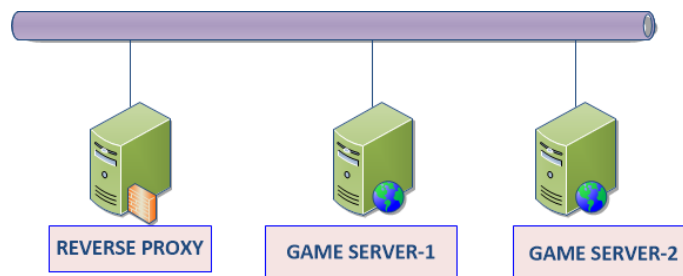


Figure 11. Reverse proxy server.

E. ADDITIONAL SECURITY MEASURES

1. Encryption

Data traffic between the reverse proxy server and its clients will be encrypted using SSL/TLS. The outcome traffic will be HTTPS which is, in short, an HTTP session encrypted with SSL/TLS. The only port that will be open is port 443 (DISA, 2006).

It is also determined that encrypting only the communication between server clients and the reverse proxy server would be enough. Other than that the traffic between the game servers and database server will not be encrypted to keep performance levels high.

2. Authentication

Another fact about SSL/TLS encryption is that it will provide server/client authentication by exchanging digital signatures between the reverse proxy server and its client. When a gamer wants to access a game site SSL/TLS authentication will occur, and the browser of the gamer will handshake with the server via the server's digital signature. This digital signature can be provided in two ways: creating and getting it verified from a third party Certificate Authority (CA) or by creating and signing itself (Tracy, 2007). Here, the digital signature will not be provided by a third party, it will be created by the server itself.

User authentication with passwords and user names will be provided by the gaming software.

IV. IMPLEMENTATION

A. INTRODUCTION

In the previous chapter we have mentioned facts about the network design and topology and explained the design process including the proper locations of network elements, necessary services to be used by those elements and security concepts to be applied to provide confidentiality, integrity and availability. We have also validated the deployment of separate subnets and security components according to the necessary documents of DISA-DoD.

In this chapter we will explain how we implemented our system with a slight difference from the initial design. We will show the details, such as IP addressing-subnetting, server operating system selections, proper usage of the necessary network services, firewall rules and IDS rules for efficient network security. At the end we will perform a network penetration test with popular attack tools.

We will not discuss the troubles we came across while implementing the firewall and IDS rules although there were many, and we spent a considerable amount of time troubleshooting them. In particular, Redhat license expirations, conflicts between firewall rules and reverse proxy encryption issues were some of those troubles that we needed to address.

B. GENERAL NETWORK INFORMATION

1. Network Topology Implementation



Figure 12. Subnets in the CSFV Network.

Subnetting is necessary in the networks to maintain security. Our network elements are distributed through three subnets (Figure 12), which are 172.20.104.0, 10.0.0.0 and 192.168.0.0. While nps.edu maintains a subnet of 255.255.0.0 for the 172.20.104.0 network, the author used 255.255.255.0 subnet both for 10.0.0.0 and 192.168.0.0 networks. The main idea behind having separate subnets is to provide security in depth and to have a security-enhanced network architecture.

After creating the initial design for the CSFV network architecture we decided to make some small changes on the first design. The first of those changes is to remove the border router and replace it with the reverse proxy.(Figure 13) With this change the landing machine of the network becomes the reverse proxy, which also runs a firewall on itself. By moving the reverse proxy to a different network than the game servers, we aim to avoid a possible attack scenario that can compromise the game servers by estimating their IP address range. Because those servers will be on separate networks, the attacker will not easily use a network scanner (such as nmap) and get access to the game servers.

The second change to the initial design is to configure the second IDS as a host-based IDS, rather than a network-based IDS. With this change our goal is to run the first IDS to detect network-based attacks such as man-in-the-middle, packet sniffing and network scanning, whereas the second IDS is to be run on a server (database server) and detect the attacks, such as DDoS and password attacks.

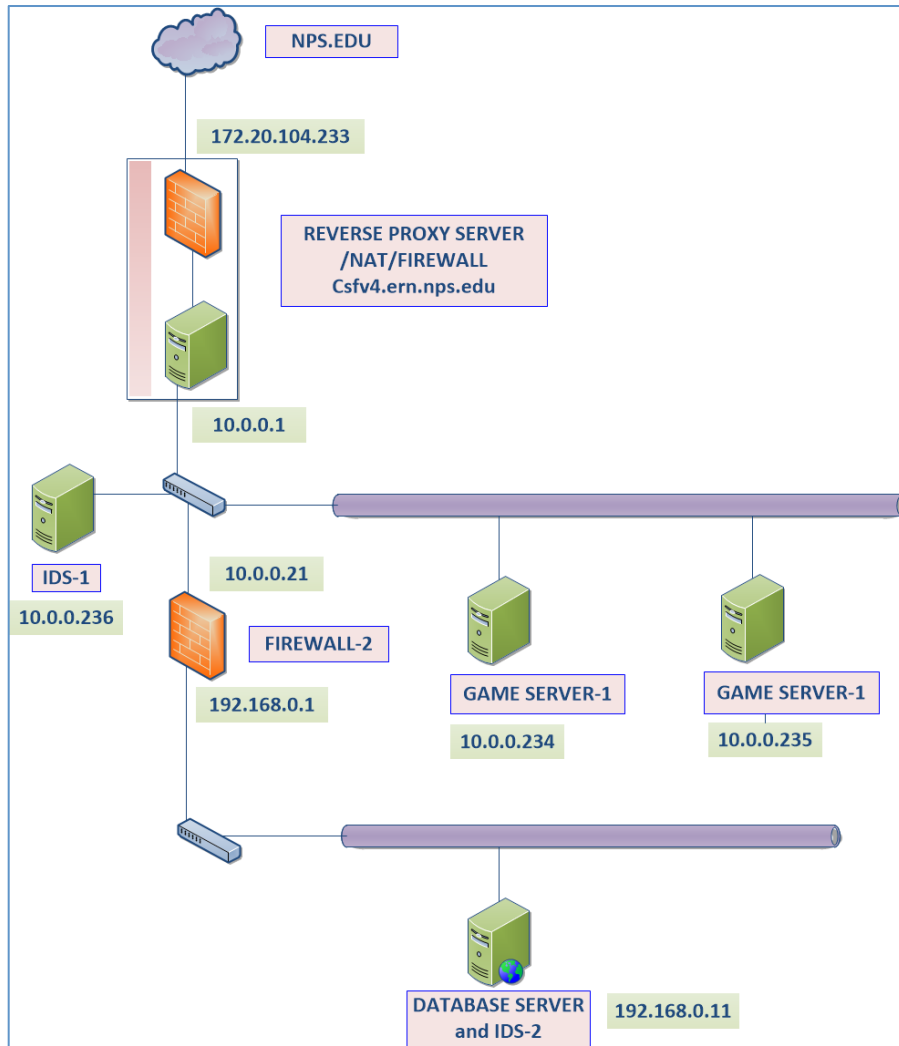


Figure 13. CSFV network implementation.

2. Implementation of Servers

There are a total of four servers, two IDS machines and two firewalls in the CSFV network. Each of them is either virtualized on a physical server or not virtualized, depending on its usage. For example, the reverse proxy server stays at the entrance of the network as a border router and forwards traffic back and forth. Because it also runs a firewall on it, we gave it a separate non-virtualized machine. On the other hand, the backend servers, which we use as game servers, run on a virtualized environment called VMware ESXi.

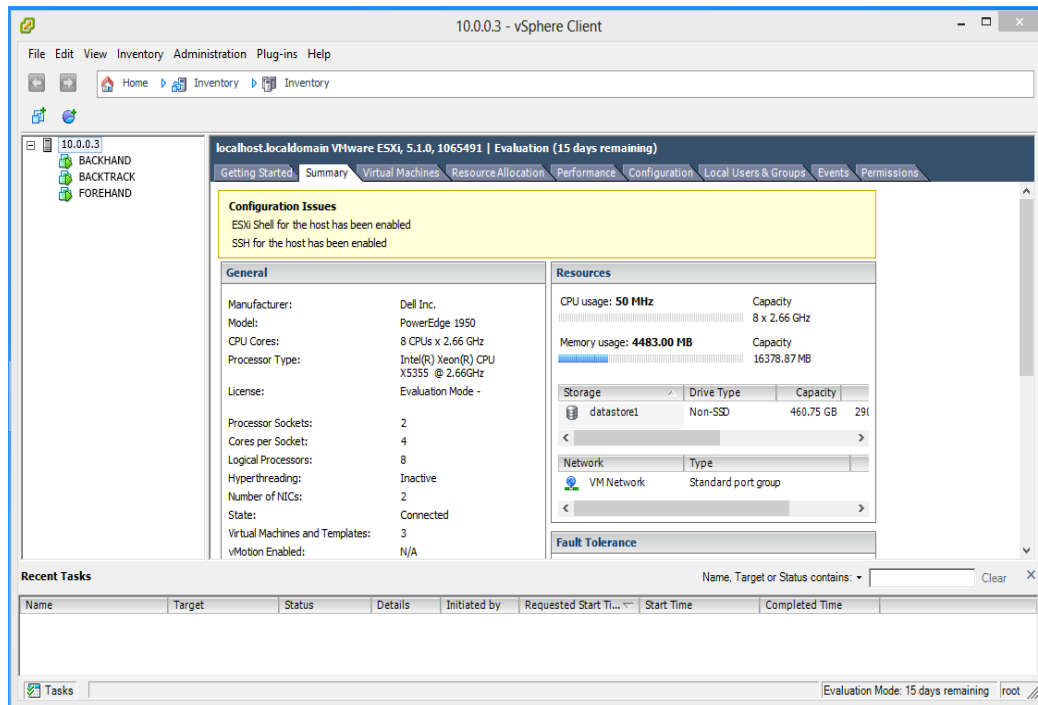
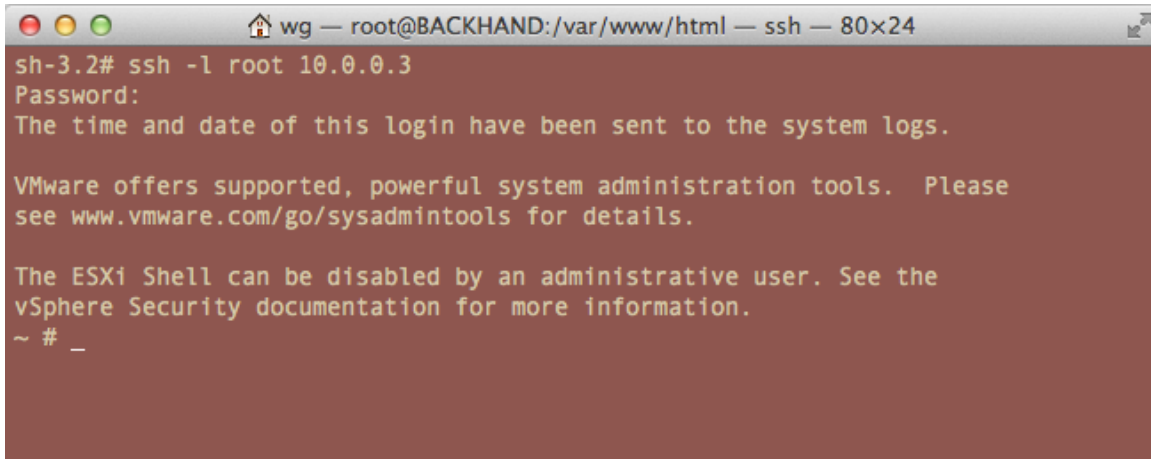


Figure 14. VMware ESXi vSphere Client.

We used VMware ESXi as the bare-metal operating system to virtualize the game servers and to make the network more efficient. As DISA STIG about VMware ESXi 5 requests, we installed and maintained the server and database operating systems through ESXi shell and the software called vSphere Client (Figure 14).

In this process the usage of ESXi shell was very helpful because it can be managed by connecting through a secure shell (Figure 15). ESXi shell is useful because it does not need a direct connection with the ESXi virtualization server. It is enough to have a remote connection to reverse proxy and use secure shell to access the virtualization server.



```
sh-3.2# ssh -l root 10.0.0.3
Password:
The time and date of this login have been sent to the system logs.

VMware offers supported, powerful system administration tools. Please
see www.vmware.com/go/sysadmintools for details.

The ESXi Shell can be disabled by an administrative user. See the
vSphere Security documentation for more information.
~ # _
```

Figure 15. VMware ESXi command shell.

VIRTUALIZED SYSTEMS ON CSFV NETWORK				
VM NAME	ROLE	HOSTNAME	IP ADDRESS	OPERATING SYSTEM
ESXi 1	WEB SERVER	csfv5	10.0.0.234	RED HAT ENTERPRISE LINUX 6.4
ESXi 1	WEB SERVER	csfv6	10.0.0.235	RED HAT ENTERPRISE LINUX 6.4
ESXi 2	IDS - 1	swing	10.0.0.236	RED HAT ENTERPRISE LINUX 6.4
ESXi 3	DATABASE SERVER and IDS-2	csfv7	192.168.0.11	RED HAT ENTERPRISE LINUX 6.4

Table 2. Virtualized systems.

As it is seen from the Table 2, there are three bare-metal virtualization operating systems in the network. ESXi-2 runs the game servers at 10.0.0.0 network, ESXi-1 runs the first IDS server at 10.0.0.0 network and ESXi-3 runs the database server and the second IDS server at 192.168.0.0 network.

The non-virtualized systems (Table 3) consist of a reverse proxy server and firewall server, which run on the CentOS 6.4 operating system. They use CentOS because the first CSFV systems, which would run on Amazon cloud architecture, EC2, would have CentOS as a server operating system. CentOS shares the same source code as Red

Hat, and the main difference between them is that the necessary update packages are publicly open source (Red Hat Enterprise Linux Installation Guide, 2013).

NON-VIRTUALIZED SYSTEMS ON CSFV NETWORK			
ROLE	HOSTNAME	IP ADDRESS	OPERATING SYSTEM
REVERSE PROXY + FIREWALL	csfv4	172.20.104.233	CENTOS 6.4
FIREWALL-2	second chance	10.0.0.21	CENTOS 6.4

Table 3. Non-virtualized Systems.

Additionally, we downloaded and installed the necessary software packages from Red Hat repository to run the games appropriately. The necessary coordination was done with the contactor from TopCoder (CSFV PI Meeting, 2013). In total, 16 packages are displayed in this chapter, and they are installed particularly to run the games. The remaining 422 packages, which are necessary for any other Linux box, are shown in the Appendix.

collectd.x86_64	4.10.9-1.el6	@epel
epel-release.noarch	6-8	@epel
gccxml.x86_64	0.9.0-0.12.20120309.el6	@epel
ius-release.noarch	1.0-11.ius.el6	@ius
joe.x86_64	3.7-4.el6	@epel
kernel.x86_64	2.6.32- 220.17.1.el6.centos.plus	@centosplus
kernel.x86_64	2.6.32- 358.6.1.el6.centos.plus	@centosplus
kernel-devel.x86_64	2.6.32- 220.17.1.el6.centos.plus	@centosplus
kernel-devel.x86_64	2.6.32- 358.6.1.el6.centos.plus	@centosplus
kernel-firmware.noarch	2.6.32- 358.6.1.el6.centos.plus	@centosplus
kernel-headers.x86_64	2.6.32- 358.6.1.el6.centos.plus	@centosplus
links.x86_64	1:2.2-12.el6	@epel
mongo-10gen.x86_64	2.4.3-mongodb_1	@10gen
mongo-10gen-server.x86_64	2.4.3-mongodb_1	@10gen
nginx.x86_64	1.4.1-1.el6.ngx	@nginx
xmlstarlet.x86_64	1.3.1-1.el6	@epel

Table 4. Necessary packages from CentOS repository

3. Running Firewall Rules

The CSFV network uses the Linux IPtables application that is embedded on the Red Hat/CentOS operating systems to define firewall rules for the firewall servers. We chose to use the Linux IPtables feature because it does not need any additional hardware or software, which would incur additional cost; it is a powerful tool used world-wide, and it is flexible enough to give the system administrator a wide area of rule options.

IPtables operates by using the “TABLE” concept, which has “CHAINS”. The two main tables are “NAT” and “FILTER.” The NAT table has PREROUTING, POSTROUTING and OUTPUT chains. On the other hand, the FILTER table includes

INPUT, OUTPUT and FORWARD chains. IPtables also has the feature of user-defined rules.

Our IPtables rules include two tables. The “NAT” table provides the network with Network Address Translation (NAT) to access outside of the network. It is also a tool to enhance security and availability of the network. The NAT rules make the servers inside our network access the IP range outside of the CSFV network by getting different IP addresses.

The FILTER table does most of the work in the network. It accepts certain IP addresses and ports, drops unwanted IP traffic for security or performance reasons and logs the IP packets, which were dropped for further inspection.

Our first firewall (csfv4.ern.nps.edu) has the rules to accept incoming valid packets. It is set to allow the incoming traffic to the server’s 22 and 443 ports.

```
-A INPUT -s 0/0 -i eth0 -d 172.20.104.233 -p TCP --dport 443 -j ACCEPT
-A INPUT -s 0/0 -i eth0 -d 172.20.104.233 -p TCP --dport 22 -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -p tcp -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -s 172.20.104.233 -o eth0 -d 0/0 -p tcp --sport 22 -j ACCEPT
-A OUTPUT -p tcp -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -p tcp -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -p tcp --dport 22 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Csfv4.ern.nps.edu has the rules to drop the invalid packets, spoofed packets, XMAS scan packets and NULL scan packets, which are techniques for port scanning.

```
-A INPUT -m state --state INVALID -j DROP
-A FORWARD -m state --state INVALID -j DROP
-A OUTPUT -m state --state INVALID -j DROP
-A INPUT -s 10.0.0.0/24 -i eth0 -d 172.20.104.233 -p tcp -
j DROP
-A INPUT -s 192.168.0.0/24 -i eth0 -d 172.20.104.233 -p
tcp -j DROP
-A INPUT -s 10.0.0.0/24 -i eth0 -d 172.20.104.233 -p udp -
j DROP
-A INPUT -s 192.168.0.0/24 -i eth0 -d 172.20.104.233 -p
udp -j DROP
-A INPUT -p tcp --tcp-flags ALL ALL -j DROP
-A INPUT -p tcp --tcp-flags ALL NONE -j DROP
```

We used also DROP policies to define a default rule for packets.

```
-P INPUT DROP
-P OUTPUT DROP
-P FORWARD DROP
```

For logging dropped packages:

```
-A INPUT -m limit --limit 15/minute -j LOG --log-level 4 --
log-prefix "DROPPED PACKETS_FIRUZ: "
```

The second firewall, “Second Chance,” has the rules for allowing the traffic between the game servers and the database server and drop else.

Rules to accept necessary packets:

```
-A INPUT -s 10.0.0.1 -i eth0 -d 10.0.0.21 -p TCP --dport 22 -j ACCEPT
-A OUTPUT -s 10.0.0.21 -o eth0 -d 10.0.0.1 -p tcp --sport 22 -j ACCEPT
-A OUTPUT -s 10.0.0.21 -o eth1 -d 192.168.0.11 -p TCP --sport 22 -j ACCEPT
-A INPUT -s 192.168.0.11 -i eth1 -d 10.0.0.21 -p tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -p tcp -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -p tcp -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -p tcp --dport 22 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -s 10.0.0.234 -d 192.168.0.11 -p tcp --dport 27017 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -s 10.0.0.235 -d 192.168.0.11 -p tcp --dport 27017 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A OUTPUT -p icmp -j ACCEPT
```

The rule to log malicious or unnecessary traffic:

```
-A INPUT -m limit --limit 15/minute -j LOG --log-level 4 --log-prefix "DROPPED PACKETS_FIRUZ: "
```

Rules to drop invalid packets:

```
-A INPUT -m state --state INVALID -j DROP
-A FORWARD -m state --state INVALID -j DROP
-A OUTPUT -m state --state INVALID -j DROP
-P INPUT DROP
-P OUTPUT DROP
-P FORWARD DROP
```

4. Reverse Proxy Deployment

As mentioned previously, the reverse proxy is implemented outside of the 10.0.0.0 network. It is done this way to provide more security by letting game servers stay on a subnet other than the reverse proxy.

An Apache web server was configured to provide the reverse proxy function. When a user types “csfv4.ern.nps.edu” in his browser, our web server directs this request to one of the game servers and gathers the data by load balancing the user requests.

The Apache configuration rules, which reside at /var/httpd/conf/httpd.conf, enable the Apache server to transfer the files from game servers to the users and to load balance the requests. We added these rules to the configuration file of Apache on csfv4.ern.nps.edu:

```
<IfModule mod_proxy.c>
ProxyRequests Off
<Proxy *>
    Order deny,allow
    Allow from all
</Proxy>
#The name of the load balancer is "mycluster" which will
#distribute all requests between 10.0.0.234 and 10.0.0.235
<Proxy balancer://mycluster>
BalancerMember http://10.0.0.234:80
BalancerMember http://10.0.0.235:80
</Proxy>
ProxyPass / balancer://mycluster
ProxyPreserveHost On
ProxyVia On
SSLProxyEngine on
ProxyPass / http://10.0.0.234:80
ProxyPass / http://10.0.0.235:80
ProxyPassReverse / http://10.0.0.234:80
ProxyPassReverse / http://10.0.0.235:80
```

5. IDS Deployment

As we mentioned earlier, firewall rules will operate at layer 4, inspecting the socket pairs. Because we did not use a deep packet inspection firewall to inspect the packet traffic at the application level, we should use an IDS device to watch the traffic above layer 4. We chose Snort as our network monitoring and IDS solution.

Snort is an open source tool, which can serve as a sniffer, logger or a network-based intrusion detection system. There are many organizations around the world that use SNORT for intrusion detection. It is also used for protecting the DREN network on the NPS campus.

Snort has some rule sets as the default. In addition, other necessary packages can be installed or user-defined packages can be created (Alder, 2004). We used Snort's predefined packages for the CSFV network. Being deployed both on the 10.0.0.0 subnet and 192.168.0.0 subnet, Snort will use a wide range of rule sets to detect malicious traffic, some of which are bad data traffic, exploits about mysql, sql injection, web attacks, DDoS, flash, chat and browser vulnerabilities. The Snort rules on CSFV network include:

```
# include $SO_RULE_PATH/bad-traffic.rules
# include $SO_RULE_PATH/chat.rules
# include $SO_RULE_PATH/dos.rules
# include $SO_RULE_PATH/exploit.rules
# include $SO_RULE_PATH/icmp.rules
# include $SO_RULE_PATH/imap.rules
# include $SO_RULE_PATH/misc.rules
# include $SO_RULE_PATH/multimedia.rules
# include $SO_RULE_PATH/netbios.rules
# include $SO_RULE_PATH/nnntp.rules
# include $SO_RULE_PATH/p2p.rules
# include $SO_RULE_PATH/smtp.rules
# include $SO_RULE_PATH/snmp.rules
# include $SO_RULE_PATH/specific-threats.rules
# include $SO_RULE_PATH/web-activex.rules
# include $SO_RULE_PATH/web-client.rules
# include $SO_RULE_PATH/web-iis.rules
# include $SO_RULE_PATH/web-misc.rules
```

6. Data Encryption

Data encryption is implemented for the data in motion on the CSFV network. As previously stated SSL protocol is used to encrypt the data flowing between the game players and the reverse proxy server. Because port 80 would be closed on the firewall of csfv4.ern.nps.edu, all data traffic would go through port 443, which is used by HTTPS. The data traffic on the CSFV network, however, will be unencrypted going through port 80. Here, the idea is that if we encrypt the moving data inside the network it needs to be decrypted on the reverse proxy server, re-encrypted and served to the clients. If a malicious user can get to the reverse proxy server as the man-in-the-middle he can also process the data, and the security would be worthless (Kew, 2003). Thus; we use SSL encryption between clients and the reverse proxy server.

The necessary SSL module (mod_ssl) for the CentOS operating system to encrypt the data traffic was downloaded from the CentOS repository (Figure 16). It runs as an Apache httpd module dependent.



```
wg — root@csfv4:~ — ssh — 83x40
Installed Packages
Name      : mod_ssl
Arch      : x86_64
Epoch    : 1
Version   : 2.2.15
Release   : 28.el6.centos
Size      : 183 k
Repo      : installed
From repo : updates
Summary   : SSL/TLS module for the Apache HTTP Server
URL       : http://httpd.apache.org/
License   : ASL 2.0
Description : The mod_ssl module provides strong cryptography for the Apache Web
              : server via the Secure Sockets Layer (SSL) and Transport Layer
              : Security (TLS) protocols.
[root@csfv4 ~]#
```

Figure 16. The SSL module for CentOS .

7. System Validation via Penetration Testing Tools

To test the capability of Snort, we directed some attacks on the 10.0.0.0 subnet. The first attack tool we used is Nmap, which is widely used to map networks by scanning ports and gathering port information. We scanned the 10.0.0.0 network range with Nmap and triggered Snort.

The second tool we used to test Snort is Low Orbit Ion Cannon (LOIC). This tool sends thousands of tcp, udp or http packets to make DDoS attacks on specific IP addresses or web sites. It is widely used on Internet to attack targets.

We used LOIC to perform attacks on the second game server's (csfv6-10.0.0.235) port 80 using udp packets (Figure 17). This attack is also caught by Snort (Table 5).

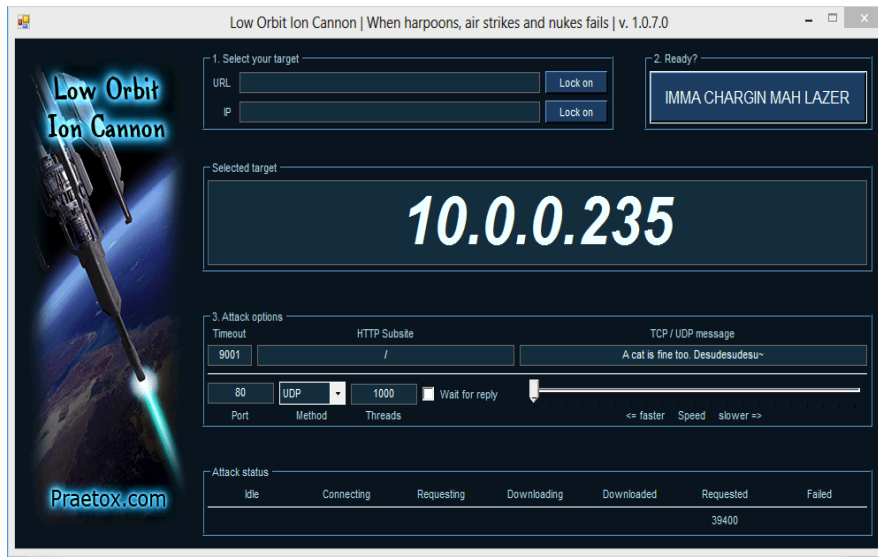


Figure 17. Attacking on the game server via LOIC.

	SNORT ALERTS
1	<pre> [**] [1:100000160:2] COMMUNITY SIP TCP/IP message flooding directed to SIP proxy [**] [Classification: Attempted Denial of Service] [Priority: 2] 08/04-04:54:48.095433 10.0.0.235:593 -> 10.0.0.51:34225 TCP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:40 DF ***A*R** Seq: 0x0 Ack: 0xF97CB988 Win: 0x0 TcpLen: 20 </pre>
2	<pre> [**] [1:100000160:2] COMMUNITY SIP TCP/IP message flooding directed to SIP proxy [**] [Classification: Attempted Denial of Service] [Priority: 2] 08/04-04:54:48.095960 10.0.0.51:34225 -> 10.0.0.235:1084 TCP TTL:55 TOS:0x0 ID:23251 IpLen:20 DgmLen:44 *****S* Seq: 0xF97CB987 Ack: 0x0 Win: 0x400 TcpLen: 24 TCP Options (1) => MSS: 1460 </pre>
3	<pre> [**] [116:59:1] (snort_decoder): Tcp Window Scale Option found with length > 14 [**] [Priority: 3] 08/04-04:54:49.016109 10.0.0.51:60511 -> 10.0.0.235:1 TCP TTL:52 TOS:0x0 ID:51460 IpLen:20 DgmLen:60 **U*P**F Seq: 0x6BC901C6 Ack: 0xBAB16756 Win: 0xFFFF TcpLen: 40 UrgPtr: 0x0 TCP Options (5) => WS: 15 NOP MSS: 265 TS: 4294967295 0 SackOK </pre>
4	<pre> [**] [122:1:0] (portscan) TCP Portscan [**] [Priority: 3] 08/04-04:55:44.906805 10.0.0.51 -> 10.0.0.2 PROTO:255 TTL:0 TOS:0x0 ID:27082 IpLen:20 DgmLen:160 DF </pre>
5	<pre> [**] [1:100000160:2] COMMUNITY SIP TCP/IP message flooding directed to SIP proxy [**] [Classification: Attempted Denial of Service] [Priority: 2] 08/04-04:55:47.361994 10.0.0.51:40883 -> 10.0.0.21:49157 TCP TTL:53 TOS:0x0 ID:59640 IpLen:20 DgmLen:44 *****S* Seq: 0x4C1BBD2 Ack: 0x0 Win: 0x400 TcpLen: 24 TCP Options (1) => MSS: 1460 </pre>

Table 5. SNORT alerts

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION

A. SUMMARY AND CONCLUSION

Manual formal software verification is an expensive and time-consuming process. The verification of military software is currently performed by highly skilled analysts (Dean, 2013). To reduce the high costs of the formal verification, DARPA started a Crowd-Sourced Formal Verification (CSFV) program in 2011. The goal of the program is to encourage as many people as possible to participate in this verification process by embedding some of the verification logics into computer games that are fun to play.

In this study the CSFV network is designed and implemented according to the common security practices, necessary security measures against possible attacks, and DISA STIGs to configure network components. After validation and verification steps we observe that the system is working well with all its security elements, and it can be trusted to deploy on a secure DoD network. We recommend that the DoD install and experiment with our CSFV network prototype on its systems.

The main goal of this thesis study is to design and prototype a secure and robust infrastructure for CSFV games. After going through a review of the literature and carrying out the design and implementation steps, we conclude that our CSFV system prototype provides these key features:

IP address filtering and NAT: Our CSFV system prototype provides IP address and port filtering with its firewall servers. First firewall rules are set to prevent network attacks by dropping spoofed, invalid and malicious packets, such as XMAS and NULL scan packets. Also the first firewall logs those malicious packets. Furthermore this firewall server provides network address translation rules to use a different set of IP addresses from those used by the external network. Our second firewall, on the other hand, allows communication only between the game servers and the database server. This firewall provides a NAT solution as well.

Network monitoring and intrusion detection: We implemented SNORT as an IDS on both 10.0.0.0 and 192.168.0.0 subnets to monitor the network activity. SNORT

uses a wide range of rule sets to detect malicious traffic, including sql injection, web attacks, DDoS, flash, and browser vulnerabilities. In Chapter IV we tested the network security levels with common attack/scan tools.

Further security for game servers with a reverse proxy server: By deploying a reverse proxy server outside the CSFV network, we aimed to avoid direct communication between game players and the CSFV game servers. In our CSFV network prototype the reverse proxy server hands over the data from the game servers to the game players. In this way we tried to reduce the risk of compromise of any game servers. The reverse proxy also load balanced the game servers depending on the number of game players.

Secure data transfer over SSL: All the game data flowing between the reverse proxy server and game players are encrypted by the SSL protocol, and they use port 443.

B. FUTURE WORK AND CONSIDERATIONS

This section presents further methods of increasing the security levels of the CSFV network prototype:

- Currently game servers on the CSFV network are kept on the same subnet for the sake of network performance and simplicity. However, to provide even more security, each game server can be placed in separate subnets to prevent simultaneous compromise of the game servers.
- There is a single database to store both game data and user data. A separate database can be designated only for user data to create another security layer for user privacy. Game data would then stay on a different database.
- Network based attacks can be augmented including social engineering attacks and other network penetration tools.
- Alternative scenarios can be created to isolate game servers from the reverse proxy in case of a compromise of the reverse proxy server.

APPENDIX

The Necessary Yum Packages to Run CSFV Games:

MAKEDEV.x86_64	3.24-6.el6	@base
abrt.x86_64	2.0.8-15.el6.centos	@base
abrt-addon-kerneloops.x86_64	2.0.8-15.el6.centos	@base
abrt-libs.x86_64	2.0.8-15.el6.centos	@base
acl.x86_64	2.2.49-6.el6	@base
alsa-lib.x86_64	1.0.22-3.el6	@base
apr-util.x86_64	1.3.9-3.el6_0.1	@base
audit-libs.x86_64	2.2-2.el6	@base
atk.x86_64	1.28.0-2.el6	@base
autoconf.noarch	2.63-5.1.el6	@base
automake.noarch	1.11.1-4.el6	@base
avahi-libs.x86_64	0.6.25-12.el6	@base
basesystem.noarch	10.0-4.el6	@base
bash.x86_64	4.1.2-14.el6	@base
bc.x86_64	1.06.95-1.el6	@base
binutils.x86_64	2.20.51.0.2-5.36.el6	@base
bison.x86_64	2.4.1-5.el6	@base
btparser.x86_64	0.17-1.el6	@base
bzip2.x86_64	1.0.5-7.el6_0	@base
bzip2-libs.x86_64	1.0.5-7.el6_0	@base
ca-certificates.noarch	2010.63-3.el6_1.5	@base
cairo.x86_64	1.8.8-3.1.el6	@base
cdparanoia-libs.x86_64	10.2-5.1.el6	@base
centos-indexhtml.noarch	6-1.el6.centos	@base
centos-release.x86_64	6-4.el6.centos.10	@base
checkpolicy.x86_64	2.0.22-1.el6	@base
chkconfig.x86_64	1.3.49-3-2.el6	@base
cloog-ppl.x86_64	0.15.7-1.2.el6	@base
compat-gcc-34.x86_64	3.4.6-19.el6	@base
compat-gcc-34-g77.x86_64	3.4.6-19.el6	@base
compat-libf2c-34.x86_64	3.4.6-19.el6	@base
compat-libstdc++-296.i686	2.96-144.el6	@base
compat-readline5.x86_64	5.2-17.1.el6	@base
cpio.x86_64	2.10-11.el6_3	@base
cpp.x86_64	4.4.7-3.el6	@base
cracklib.x86_64	2.8.16-4.el6	@base
cracklib-dicts.x86_64	2.8.16-4.el6	@base
createrepo.noarch	0.9.9-17.el6	@base
cronie.x86_64	1.4.4-7.el6	@base
cronie-anacron.x86_64	1.4.4-7.el6	@base
crontabs.noarch	1.10-33.el6	@base
cvs.x86_64	1.11.23-15.el6	@base
cyrus-sasl.x86_64	2.1.23-13.el6_3.1	@base
cyrus-sasl-lib.x86_64	2.1.23-13.el6_3.1	@base
dash.x86_64	0.5.5.1-4.el6	@base
db4.x86_64	4.7.25-17.el6	@base
db4-utils.x86_64	4.7.25-17.el6	@base
dbus.x86_64	1:1.2.24-7.el6_3	@base
dbus-libs.x86_64	1:1.2.24-7.el6_3	@base
dejavu-fonts-common.noarch	2.30-2.el6	@base
dejavu-lgc-sans-mono-fonts.noarch	2.30-2.el6	@base
dejavu-sans-mono-fonts.noarch	2.30-2.el6	@base
deltarpm.x86_64	3.5-0.5.20090913git.el6	@base
device-mapper.x86_64	1.02.77-9.el6	@base
device-mapper-event.x86_64	1.02.77-9.el6	@base
device-mapper-event-libs.x86_64	1.02.77-9.el6	@base
device-mapper-libs.x86_64	1.02.77-9.el6	@base
device-mapper-multipath.x86_64	0.4.9-64.el6	@base
device-mapper-multipath-libs.x86_64	0.4.9-64.el6	@base
device-mapper-persistent-data.x86_64	0.1.4-1.el6	@base
dhclient.x86_64	12:4.1.1-34.P1.el6.centos	@base
dhcp-common.x86_64	12:4.1.1-34.P1.el6.centos	@base

diffutils.x86_64	2.8.1-28.el6	@base
dmraid.x86_64	1.0.0.rc16-11.el6	@base
dmraid-events.x86_64	1.0.0.rc16-11.el6	@base
dracut.noarch	004-303.el6	@base
dracut-kernel.noarch	004-303.el6	@base
e2fsprogs.x86_64	1.41.12-14.el6	@base
e2fsprogs-libs.x86_64	1.41.12-14.el6	@base
ecj.x86_64	1:3.4.2-6.el6	@base
ed.x86_64	1.1-3.3.el6	@base
elfutils.x86_64	0.152-1.el6	@base
elfutils-libelf.x86_64	0.152-1.el6	@base
elfutils-libs.x86_64	0.152-1.el6	@base
ethtool.x86_64	2:3.5-1.el6	@base
file.x86_64	5.04-15.el6	@base
file-libs.x86_64	5.04-15.el6	@base
filesystem.x86_64	2.4.30-3.el6	@base
findutils.x86_64	1:4.4.2-6.el6	@base
fipscheck.x86_64	1.2.0-7.el6	@base
fipscheck-lib.x86_64	1.2.0-7.el6	@base
flex.x86_64	2.5.35-8.el6	@base
fontconfig.x86_64	2.8.0-3.el6	@base
fontpackages-filesystem.noarch	1.41-1.1.el6	@base
foomatic.x86_64	4.0.4-1.el6_1.1	@base
foomatic-db.noarch	4.0-7.20091126.el6	@base
foomatic-db-filesystem.noarch	4.0-7.20091126.el6	@base
foomatic-db-ppds.noarch	4.0-7.20091126.el6	@base
gamin.x86_64	0.1.10-9.el6	@base
gawk.x86_64	3.1.7-10.el6	@base
gcc.x86_64	4.4.7-3.el6	@base
gcc-c++.x86_64	4.4.7-3.el6	@base
gcc-gfortran.x86_64	4.4.7-3.el6	@base
gcc-gnat.x86_64	4.4.7-3.el6	@base
gcc-java.x86_64	4.4.7-3.el6	@base
gcc-objc.x86_64	4.4.7-3.el6	@base
gcc-objc++.x86_64	4.4.7-3.el6	@base
gdb.x86_64	7.2-60.el6	@base
gdbm.x86_64	1.8.0-36.el6	@base
gettext.x86_64	0.17-16.el6	@base
ghostscript-fonts.noarch	5.50-23.1.el6	@base
glib2.x86_64	2.22.5-7.el6	@base
glibc.i686	2.12-1.107.el6	@base
glibc.x86_64	2.12-1.107.el6	@base
glibc-common.x86_64	2.12-1.107.el6	@base
glibc-devel.x86_64	2.12-1.107.el6	@base
glibc-headers.x86_64	2.12-1.107.el6	@base
gnupg2.x86_64	2.0.14-4.el6	@base
gpgme.x86_64	1.1.8-3.el6	@base
gpm-libs.x86_64	1.20.6-12.el6	@base
grep.x86_64	2.6.3-3.el6	@base
groff.x86_64	1.18.1.4-21.el6	@base
grubby.x86_64	7.0.15-3.el6	@base
gststreamer.x86_64	0.10.29-1.el6	@base
gststreamer-plugins-base.x86_64	0.10.29-2.el6	@base
gststreamer-tools.x86_64	0.10.29-1.el6	@base
gtk2.x86_64	2.18.9-12.el6	@base
gzip.x86_64	1.3.12-18.el6	@base
hicolor-icon-theme.noarch	0.11-1.1.el6	@base
hwdata.noarch	0.233-7.9.el6	@base
info.x86_64	4.13a-8.el6	@base
iproute.x86_64	2.6.32-23.el6	@base
iptables.x86_64	1.4.7-9.el6	@base
iputils.x86_64	20071127-16.el6	@base
iso-codes.noarch	3.16-2.el6	@base
java-1.5.0-gcj.x86_64	1.5.0.0-29.1.el6	@base
java_cup.x86_64	1:0.10k-5.el6	@base
jpackage-utils.noarch	1.7.5-3.12.el6	@base
kbd.x86_64	1.15-11.el6	@base
kbd-misc.noarch	1.15-11.el6	@base
keyutils-libs.x86_64	1.4-4.el6	@base
kpartx.x86_64	0.4.9-64.el6	@base
lcms-libs.x86_64	1.19-1.el6	@base

less.x86_64	436-10.el6	@base
libICE.x86_64	1.0.6-1.el6	@base
libSM.x86_64	1.2.1-2.el6	@base
libX11.x86_64	1.5.0-4.el6	@base
libX11-common.noarch	1.5.0-4.el6	@base
libXau.x86_64	1.0.6-4.el6	@base
libXcomposite.x86_64	0.4.3-4.el6	@base
libXcursor.x86_64	1.1.13-2.el6	@base
libXdamage.x86_64	1.1.3-4.el6	@base
libXext.x86_64	1.3.1-2.el6	@base
libXfixes.x86_64	5.0-3.el6	@base
libXfont.x86_64	1.4.5-2.el6	@base
libXft.x86_64	2.3.1-2.el6	@base
libXi.x86_64	1.6.1-3.el6	@base
libXinerama.x86_64	1.1.2-2.el6	@base
libXrandr.x86_64	1.4.0-1.el6	@base
libXrender.x86_64	0.9.7-2.el6	@base
libXt.x86_64	1.1.3-1.el6	@base
libXtst.x86_64	1.2.1-2.el6	@base
libXv.x86_64	1.0.7-2.el6	@base
libXxf86vm.x86_64	1.1.2-2.el6	@base
libacl.x86_64	2.2.49-6.el6	@base
libaio.x86_64	0.3.107-10.el6	@base
libart_lgpl.x86_64	2.3.20-5.1.el6	@base
libattr.x86_64	2.4.44-7.el6	@base
libcap.x86_64	2.16-5.5.el6	@base
libcap-ng.x86_64	0.6.4-3.el6_0.1	@base
libcom_err.x86_64	1.41.12-14.el6	@base
libdrm.x86_64	2.4.39-1.el6	@base
libedit.x86_64	2.11-4.20080712cvs.1.el6	@base
libevent.x86_64	1.4.13-4.el6	@base
libffi.x86_64	3.0.5-3.2.el6	@base
libfontenc.x86_64	1.0.5-2.el6	@base
libgcc.i686	4.4.7-3.el6	@base
libgcc.x86_64	4.4.7-3.el6	@base
libgccj.x86_64	4.4.7-3.el6	@base
libgccj-devel.x86_64	4.4.7-3.el6	@base
libgfortran.x86_64	4.4.7-3.el6	@base
libgnat.x86_64	4.4.7-3.el6	@base
libgnat-devel.x86_64	4.4.7-3.el6	@base
libgomp.x86_64	4.4.7-3.el6	@base
libgpg-error.x86_64	1.7-4.el6	@base
libgudev1.x86_64	147-2.46.el6	@base
libidn.x86_64	1.18-2.el6	@base
libjpeg-turbo.x86_64	1.2.1-1.el6	@base
libmng.x86_64	1.0.10-4.1.el6	@base
libnih.x86_64	1.0.1-7.el6	@base
libobjc.x86_64	4.4.7-3.el6	@base
libogg.x86_64	2:1.1.4-2.1.el6	@base
liboil.x86_64	0.3.16-4.1.el6	@base
libpciaccess.x86_64	0.13.1-2.el6	@base
libreport.x86_64	2.0.9-15.el6.centos	@base
libreport-compat.x86_64	2.0.9-15.el6.centos	@base
libreport-plugin-kerneloops.x86_64	2.0.9-15.el6.centos	@base
libreport-plugin-reportuploader.x86_64	2.0.9-15.el6.centos	@base
libreport-plugin-rhtsupport.x86_64	2.0.9-15.el6.centos	@base
libreport-python.x86_64	2.0.9-15.el6.centos	@base
libselenium.x86_64	2.0.94-5.3.el6	@base
libselenium-utils.x86_64	2.0.94-5.3.el6	@base
libsemanage.x86_64	2.0.43-4.2.el6	@base
libsepol.x86_64	2.0.41-4.el6	@base
libss.x86_64	1.41.12-14.el6	@base
libssh2.x86_64	1.4.2-1.el6	@base
libstdc++.x86_64	4.4.7-3.el6	@base
libstdc++-devel.x86_64	4.4.7-3.el6	@base
libsysfs.x86_64	2.1.0-7.el6	@base
libtar.x86_64	1.2.11-17.el6	@base
libthai.x86_64	0.1.12-3.el6	@base
libtheora.x86_64	1:1.1.0-2.el6	@base
libtiff.x86_64	3.9.4-9.el6_3	@base
libtool.x86_64	2.2.6-15.5.el6	@base

libudev.x86_64	147-2.46.el6	@base
libusb.x86_64	0.1.12-23.el6	@base
libuser.x86_64	0.56.13-5.el6	@base
libutempter.x86_64	1.1.5-4.1.el6	@base
libvisual.x86_64	0.4.0-9.1.el6	@base
libxcb.x86_64	1.8.1-1.el6	@base
libxslt.x86_64	1.1.26-2.el6_3.1	@base
logrotate.x86_64	3.7.8-16.el6	@base
lua.x86_64	5.1.4-4.1.el6	@base
lvm2.x86_64	2.02.98-9.el6	@base
lvm2-libs.x86_64	2.02.98-9.el6	@base
lynx.x86_64	2.8.6-27.el6	@base
m2crypto.x86_64	0.20.2-9.el6	@base
m4.x86_64	1.4.13-5.el6	@base
mailcap.noarch	2.1.31-2.el6	@base
mailx.x86_64	12.4-6.el6	@base
make.x86_64	1:3.81-20.el6	@base
man.x86_64	1.6f-32.el6	@base
mcstrans.x86_64	0.3.1-4.el6	@base
memcached.x86_64	1.4.4-3.el6	@base
mercurial.x86_64	1.4-3.el6	@base
mesa-dri-drivers.x86_64	9.0-0.7.el6	@base
mesa-dri-filesystem.x86_64	9.0-0.7.el6	@base
mesa-drii-drivers.x86_64	7.11-8.el6	@base
mesa-libGL.x86_64	9.0-0.7.el6	@base
mesa-libGLU.x86_64	9.0-0.7.el6	@base
mingetty.x86_64	1.08-5.el6	@base
mlocate.x86_64	0.22.2-4.el6	@base
module-init-tools.x86_64	3.9-21.el6	@base
mpfr.x86_64	2.4.1-6.el6	@base
mutt.x86_64	5:1.5.20-2.20091214hg736b6a.el6_1.1	@base
nano.x86_64	2.0.9-7.el6	@base
ncurses.x86_64	5.7-3.20090208.el6	@base
ncurses-base.x86_64	5.7-3.20090208.el6	@base
ncurses-libs.x86_64	5.7-3.20090208.el6	@base
neon.x86_64	0.29.3-2.el6	@base
nspr.x86_64	4.9.2-1.el6	@base
nss.x86_64	3.14.0.0-12.el6	@base
nss-softokn.x86_64	3.12.9-11.el6	@base
nss-softokn-freebl.i686	3.12.9-11.el6	@base
nss-softokn-freebl.x86_64	3.12.9-11.el6	@base
nss-sysinit.x86_64	3.14.0.0-12.el6	@base
nss-tools.x86_64	3.14.0.0-12.el6	@base
nss-util.x86_64	3.14.0.0-2.el6	@base
nss_compat_oss1.x86_64	0.9.6-1.el6	@base
openjpeg-libs.x86_64	1.3-9.el6_3	@base
openssh.x86_64	5.3p1-84.1.el6	@base
openssh-askpass.x86_64	5.3p1-84.1.el6	@base
openssh-clients.x86_64	5.3p1-84.1.el6	@base
openssh-server.x86_64	5.3p1-84.1.el6	@base
pakchois.x86_64	0.4-3.2.el6	@base
pam.x86_64	1.1.1-13.el6	@base
pango.x86_64	1.28.1-7.el6_3	@base
patch.x86_64	2.6-6.el6	@base
pax.x86_64	3.4-10.1.el6	@base
pcre.x86_64	7.8-6.el6	@base
perl-Error.noarch	1:0.17015-4.el6	@base
perl-URI.noarch	1.40-2.el6	@base
pinentry.x86_64	0.7.6-6.el6	@base
pkgconfig.x86_64	1:0.23-9.1.el6	@base
plymouth.x86_64	0.8.3-27.el6.centos	@base
plymouth-core-libs.x86_64	0.8.3-27.el6.centos	@base
plymouth-scripts.x86_64	0.8.3-27.el6.centos	@base
policycoreutils.x86_64	2.0.83-19.30.el6	@base
poppler.x86_64	0.12.4-3.el6_0.1	@base
poppler-data.noarch	0.4.0-1.el6	@base
poppler-utils.x86_64	0.12.4-3.el6_0.1	@base
popt.x86_64	1.13-7.el6	@base
portreserve.x86_64	0.0.4-9.el6	@base
postfix.x86_64	2:2.6.6-2.2.el6_1	@base
ppl.x86_64	0.10.2-11.el6	@base

procps.x86_64	3.2.8-25.el6	@base
psmisc.x86_64	22.6-15.el6_0.1	@base
pth.x86_64	2.0.7-9.3.el6	@base
pygpgme.x86_64	0.1-18.20090824bZR68.el6	@base
python.x86_64	2.6.6-36.el6	@base
python-deltarpm.x86_64	3.5-0.5.20090913git.el6	@base
python-iniparse.noarch	0.3.1-2.1.el6	@base
python-libs.x86_64	2.6.6-36.el6	@base
python-pycurl.x86_64	7.19.0-8.el6	@base
python-urlgrabber.noarch	3.9.1-8.el6	@base
qt3.x86_64	3.3.8b-30.el6	@base
readline.x86_64	6.0-4.el6	@base
redhat-logos.noarch	60.0.14-12.el6.centos	@base
redhat-lsb.x86_64	4.0-7.el6.centos	@base
redhat-lsb-compat.x86_64	4.0-7.el6.centos	@base
redhat-lsb-core.x86_64	4.0-7.el6.centos	@base
redhat-lsb-graphics.x86_64	4.0-7.el6.centos	@base
redhat-lsb-printing.x86_64	4.0-7.el6.centos	@base
redhat-rpm-config.noarch	9.0.3-42.el6	@base
rootfiles.noarch	8.1-6.1.el6	@base
rpm.x86_64	4.8.0-32.el6	@base
rpm-build.x86_64	4.8.0-32.el6	@base
rpm-libs.x86_64	4.8.0-32.el6	@base
rpm-python.x86_64	4.8.0-32.el6	@base
rrdtool.x86_64	1.3.8-6.el6	@base
rrdtool-devel.x86_64	1.3.8-6.el6	@base
rrdtool-doc.x86_64	1.3.8-6.el6	@base
rrdtool-perl.x86_64	1.3.8-6.el6	@base
rrdtool-python.x86_64	1.3.8-6.el6	@base
rrdtool-ruby.x86_64	1.3.8-6.el6	@base
rrdtool-tcl.x86_64	1.3.8-6.el6	@base
rsync.x86_64	3.0.6-9.el6	@base
rsyslog.x86_64	5.8.10-6.el6	@base
rubygems.noarch	1.3.7-1.el6	@base
screen.x86_64	4.0.3-16.el6	@base
sed.x86_64	4.2.1-10.el6	@base
setup.noarch	2.8.14-20.el6	@base
sgpio.x86_64	1.2.0.10-5.el6	@base
shadow-utils.x86_64	2:4.1.4.2-13.el6	@base
sinjdoc.x86_64	0.5-9.1.el6	@base
sqlite.x86_64	3.6.20-1.el6	@base
sudo.x86_64	1.8.6p3-7.el6	@base
sysfsutils.x86_64	2.1.0-7.el6	@base
sysstat.x86_64	9.0.4-20.el6	@base
sysvinit-tools.x86_64	2.87-4.dsF.el6	@base
tar.x86_64	2:1.23-11.el6	@base
tcl.x86_64	1:8.5.7-6.el6	@base
tcp_wrappers-libs.x86_64	7.6-57.el6	@base
time.x86_64	1.7-37.1.el6	@base
tk.x86_64	1:8.5.7-5.el6	@base
tmpwatch.x86_64	2.9.16-4.el6	@base
tokyocabinet.x86_64	1.4.33-6.el6	@base
udev.x86_64	147-2.46.el6	@base
unzip.x86_64	6.0-1.el6	@base
upstart.x86_64	0.6.5-12.el6	@base
urlview.x86_64	0.9-7.el6	@base
urw-fonts.noarch	2.4-10.el6	@base
usermode.x86_64	1.102-3.el6	@base
ustr.x86_64	1.0.4-9.1.el6	@base
vim-common.x86_64	2:7.2.411-1.8.el6	@base
vim-enhanced.x86_64	2:7.2.411-1.8.el6	@base
vim-minimal.x86_64	2:7.2.411-1.8.el6	@base
wget.x86_64	1.12-1.8.el6	@base
which.x86_64	2.19-6.el6	@base
xml-common.noarch	0.6.3-32.el6	@base
xmlrpc-c.x86_64	1.16.24-1209.1840.el6	@base
xmlrpc-c-client.x86_64	1.16.24-1209.1840.el6	@base
xorg-x11-font-utils.x86_64	1:7.2-11.el6	@base
xz.x86_64	4.999.9-0.3.beta.20091007git.el6	@base
xz-libs.x86_64	4.999.9-0.3.beta.20091007git.el6	@base
xz-lzma-compat.x86_64	4.999.9-0.3.beta.20091007git.el6	@base

yajl.x86_64	1.0.7-3.el6	@base
yum.noarch	3.2.29-40.el6.centos	@base
yum-metadata-parser.x86_64	1.1.2-16.el6	@base
yum-plugin-fastestmirror.noarch	1.1.30-14.el6	@base
yum-utils.noarch	1.1.30-14.el6	@base
zip.x86_64	3.0-1.el6	@base
zlib.x86_64	1.2.3-29.el6	@base
zlib-devel.x86_64	1.2.3-29.el6	@base
apr.x86_64	1.3.9-5.el6_2	@updates
at.x86_64	3.1.10-43.el6_2.1	@updates
bind-libs.x86_64	32:9.8.2-0.17.rc1.el6_4.4	@updates
bind-utils.x86_64	32:9.8.2-0.17.rc1.el6_4.4	@updates
coreutils.x86_64	8.4-19.el6_4.1	@updates
coreutils-libs.x86_64	8.4-19.el6_4.1	@updates
cups.x86_64	1:1.4.2-50.el6_4.4	@updates
cups-libs.x86_64	1:1.4.2-50.el6_4.4	@updates
curl.x86_64	7.19.7-36.el6_4	@updates
dbus-glib.x86_64	0.86-6.el6	@updates
elinks.x86_64	0.12-0.21.pre5.el6_3	@updates
expat.x86_64	2.0.1-11.el6_2	@updates
freetype.x86_64	2.3.11-14.el6_3.1	@updates
ghostscript.x86_64	8.70-15.el6_4.1	@updates
git.x86_64	1.7.1-3.el6_4.1	@updates
gmp.x86_64	4.3.1-7.el6_2.2	@updates
gnutls.x86_64	2.8.5-10.el6_4.1	@updates
initscripts.x86_64	9.03.38-1.el6.centos.1	@updates
jasper-libs.x86_64	1.900.1-15.el6_1.1	@updates
krb5-libs.x86_64	1.10.3-10.el6_4.2	@updates
ksh.x86_64	20100621-19.el6_4.3	@updates
libblkid.x86_64	2.17.2-12.9.el6_4.3	@updates
libcurl.x86_64	7.19.7-36.el6_4	@updates
libgcrypt.x86_64	1.4.5-9.el6_2.2	@updates
libpng.x86_64	2:1.2.49-1.el6_2	@updates
libproxy.x86_64	0.3.0-4.el6_3	@updates
libproxy-bin.x86_64	0.3.0-4.el6_3	@updates
libproxy-python.x86_64	0.3.0-4.el6_3	@updates
libtasnl.x86_64	2.3-3.el6_2.1	@updates
libuuid.x86_64	2.17.2-12.9.el6_4.3	@updates
libvorbis.x86_64	1:1.2.3-4.el6_2.1	@updates
libxml2.x86_64	2.7.6-12.el6_4.1	@updates
libxml2-python.x86_64	2.7.6-12.el6_4.1	@updates
mysql-libs.x86_64	5.1.69-1.el6_4	@updates
net-tools.x86_64	1.60-110.el6_2	@updates
openldap.x86_64	2.4.23-32.el6_4.1	@updates
openssl.x86_64	1.0.0-27.el6_4.2	@updates
passwd.x86_64	0.77-4.el6_2.2	@updates
perl.x86_64	4:5.10.1-131.el6_4	@updates
perl-CGI.x86_64	3.51-131.el6_4	@updates
perl-ExtUtils-MakeMaker.x86_64	6.55-131.el6_4	@updates
perl-ExtUtils-ParseXS.x86_64	1:2.2003.0-131.el6_4	@updates
perl-Git.noarch	1.7.1-3.el6_4.1	@updates
perl-Module-Pluggable.x86_64	1:3.90-131.el6_4	@updates
perl-Pod-Escapes.x86_64	1:1.04-131.el6_4	@updates
perl-Pod-Simple.x86_64	1:3.13-131.el6_4	@updates
perl-Test-Harness.x86_64	3.17-131.el6_4	@updates
perl-Test-Simple.x86_64	0.92-131.el6_4	@updates
perl-devel.x86_64	4:5.10.1-131.el6_4	@updates
perl-libs.x86_64	4:5.10.1-131.el6_4	@updates
perl-version.x86_64	3:0.77-131.el6_4	@updates
phonon-backend-gstreamer.x86_64	1:4.6.2-26.el6_4	@updates
pixman.x86_64	0.26.2-5.el6_4	@updates
qt.x86_64	1:4.6.2-26.el6_4	@updates
qt-sqlite.x86_64	1:4.6.2-26.el6_4	@updates
qt-x11.x86_64	1:4.6.2-26.el6_4	@updates
rightscale.x86_64	5.7.14-1	installed
ruby.x86_64	1.8.7.352-10.el6_4	@updates
ruby-devel.x86_64	1.8.7.352-10.el6_4	@updates
ruby-docs.x86_64	1.8.7.352-10.el6_4	@updates
ruby-irb.x86_64	1.8.7.352-10.el6_4	@updates
ruby-libs.x86_64	1.8.7.352-10.el6_4	@updates
ruby-rdoc.x86_64	1.8.7.352-10.el6_4	@updates

ruby-ri.x86_64	1.8.7.352-10.el6_4	@updates
ruby-tcltk.x86_64	1.8.7.352-10.el6_4	@updates
subversion.x86_64	1.6.11-9.el6_4	@updates
tzdata.noarch	2013b-1.el6	@updates
util-linux-ng.x86_64	2.17.2-12.9.el6_4.3	@updates

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

- Ahn, L.V. (2006). Games with a purpose. Retrieved from <http://www.cs.cmu.edu/~biglou/ieee-gwap.pdf>
- Babbin, J., Alder, R. (2004). Snort 2.1. 3-26. Bombay, India, Schroff Publishers
- Bell, D. (2013). *The Crowdsourcing Handbook*. Newstead, Australia, Emereo Publishing
- Chou, Y. (2012). A Private Cloud Delivers IT as a Service. Retrieved from <http://blogs.technet.com/b/yungchou/>
- Dean, D. (2013). Crowd Sourced Formal Verification. Presented at CSFV PI Meeting-Conference, Skamania-WA
- Dean, D. (2013). Implementing a Prototype for CSFV. Speech at Naval Postgraduate School, Monterey, CA.
- Del Villar, A. (2013). CSFV PI meeting. CSFV Conference, Skamania, WA.
- Diffie, W., Hellman, M. (1976). New Directions in Cryptography. *Transactions on Information Theory*, IT-22, no.6. Retrieved from <http://www-ee.stanford.edu/~hellman/publications/24.pdf>
- DISA. (2006). Webserver STIG. Retrieved from <http://iase.disa.mil/stigs/>
- DISA. (2007). Network infrastructure STIG. Retrieved from <http://iase.disa.mil/stigs/>
- DISA. (2013). ESXi 5 Technology Overview. Retrieved from <http://iase.disa.mil/stigs/>
- Gabiani, G. (2011). Cloud Computing: Emerging Technology. Retrieved from <http://www.technologydoesbusiness.com/whiteboard/2011/>
- Gregory, P. (2010). *CISSP Guide to security essentials*. Boston, MA: Cengage Learning Publishing.
- Hagel, C. (2013). The Cyber Dimension to Asian Security. Speech at Shangri-La Security Dialogue Conference, Singapore.
- Halfond, W., Orso, A. (2005). Combining static analysis and runtime monitoring to counter sql-injection attacks. Workshop on Dynamic Analysis- WODA-05. Retrieved from <http://www.cc.gatech.edu/~orso/papers/halfond.orso.WODA05.pdf>
- Howe, J. (2008). *Crowdsourcing*. New York: Crown Publishing.

- Kaminski, P. (2013). *Resilient Military Systems and advanced cyber threat*. (20301). Washington, D.C.: Office of the Under Secretary of Defense for Acquisition, Technology and Logistics. Retrieved from <http://www.acq.osd.mil/dsb/reports/ResilientMilitarySystems.CyberThreat.pdf>
- Kerckhoffs, A. (1883). La cryptographie militaire. Journal des Sciences Militaires, Vol.9, pp. 5-38. Retrieved from http://www.petitcolas.net/fabien/kerckhoffs/crypto_militaire_1.pdf
- Kew, N. (2003). Running a Reverse Proxy in Apache. Retrieved from <http://www.apachetutor.org/admin/reverseproxies>
- Kuznetzky, D. (2011). *Virtualization manager's guide*. Sebastopol, CA: O'Reilly.
- Lau, W. (2011). A Comprehensive Introduction to Cloud Computing. Retrieved from <https://www.simple-talk.com/cloud/development/>
- Lynch, A. (2012). Crowdsourcing Is Booming In Asia. Retrieved from <http://techcrunch.com/2012/12/08/asias-secret-crowdsourcing-boom/>
- Mell, P., Grance, T. (2011). *The NIST Definition of Cloud Computing* (800-145). Gaithersburg-MD: National Institute of Standards and Technology. Retrieved from <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>
- Montalbano, E. (2011). DOD Looks To Make A Game Of SoftwareTesting. Retrieved from <http://www.informationweek.com/government/>
- Panetta, L. (2012). Defending The Nations Critical Infrastructure From Cyber Security Threats. Speech at Intrepid Sea, Air and Space Museum, New York, NY.
- Patrikakis, C., Masikos, M., Zouraraki, O. (2004). Distributed Denial of Service Attacks. Retrieved from http://www.cisco.com/web/about/ac123/ac147/archived_issues/
- Redhat Documentations. (2013). Red Hat Enterprise Linux Installation Guide. Retrieved from <https://access.redhat.com/site/documentation/>
- Rivera, J., Van der Meulen, R. (2013). Gartner Says Worldwide Security Market to Grow 8.7 Percent in 2013. Retrieved from <http://www.gartner.com/newsroom/id/2512215>.
- Oxenhandler, D., (2003). Designing a Secure Local Area Network. Retrieved from <http://www.sans.org/reading-room/whitepapers/bestprac/>
- Scarfone, K., Souppaya and M., Hoffman, P. (2011). *Guide to security for full virtualization technologies*. Gaithersburg-MD: National Institute of

- Standards and Technology 1-5. Retrieved from
<http://csrc.nist.gov/publications/nistpubs/800-125/SP800-125-final.pdf>
- Shilovitsky, O. (2013). Will Enterprise PLM Embrace Public Cloud. Retrieved from
<http://beyondplm.com/2013/02/15/will-enterprise-plm-embrace-hybrid-cloud/>
- Smoot, S., Tan, N. (2012). *Private Cloud Computing*. Waltham, MA: Morgan Kaufmann Publishing.
- Stallings, W., Brown, L. (2008). *Computer security, principles and practice*. Upper Saddle River, NJ: Prentice Hall.
- Department of Defense. (July, 2012). DoD cloud computing strategy [Memorandum]. Washington, D.C. Retrieved from <http://www.defense.gov/news/dodcloudcomputingstrategy.pdf>
- Tanenbaum, A. (2003). *Computer Networks*. Upper Saddle River: NJ, Prentice Hall.
- Thomas, G. (2013). Windows 8 Hyper-Visor. Retrieved from
<http://www.computerperformance.co.uk/win8/>
- Tracy, M. (2007). *NIST Guidelines On Securing Public Web Servers*. Gaithersburg-MD: National Institute of Standards and Technology. Retrieved from
<http://csrc.nist.gov/publications/nistpubs/800-44-ver2/SP800-44v2.pdf>
- Winkler, V. (2011). *Securing the Cloud: Cloud Computer Security Techniques and Tactics*. Waltham, MA: Syngress Publishing.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California